EXHIBIT T

EX. T

(PORRAS DEPOSITION EXCERPTS)

ENTIRE EXHIBIT REDACTED

EXHIBIT U

EX. U

(VALDES DEPOSITION EXCERPTS)

ENTIRE EXHIBIT REDACTED

EXHIBIT V

EX. V

(KESIDIS DEPOSITION EXCERPTS)

ENTIRE EXHIBIT REDACTED

EXHIBIT W

US006321338B1

(12) **United States Patent**
Porras et al.

(10) **Patent No.: US 6,321,338 B1**
(45) **Date of Patent: Nov. 20, 2001**

(54) **NETWORK SURVEILLANCE**

(75) Inventors: **Phillip A. Porras**, Mountain View; **Alfonso Valdes**, San Carlos, both of CA (US)

(73) Assignee: **SRI International**, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/188,739**

(22) Filed: **Nov. 9, 1998**

(51) **Int. Cl.**[7] **G06F 11/30; G06F 12/14**
(52) **U.S. Cl.** **713/201; 709/224**
(58) **Field of Search** **713/201; 709/224**

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,672,609	6/1987	Humphrey et al.	371/21
4,773,028	9/1988	Tallman	364/550
5,210,704	5/1993	Husseiny	364/551.01
5,539,659 *	7/1996	McKee et al.	709/224
5,557,742	9/1996	Smaha et al.	395/186
5,706,210 *	1/1998	Kumano et al.	709/224
5,790,799 *	8/1998	Mogul	709/224
5,974,237 *	10/1999	Shurmer et al.	709/224
6,009,467 *	12/1999	Ratcliff et al.	709/224

OTHER PUBLICATIONS

Debar et al., "A Neural Network Component for an Intrusion Detection System," © 1992 IEEE.
Denning et al., "Prototype IDES: A Real-Time Intrusion-Detection Expert System," SRI Project ECU 7508, SRI International, MenloPark, California, Aug. 1987.
Denning et al., "Requirements and Model For IDES—A Real-Time Intrusion-Detection Expert System," SRI Project 6169, SRI International, Menlo Park, CA, Aug. 1985.
Denning, "An Intrusion-Detection Model," SRI International, Menlo Park, CA, Technical Report CSL-149, Nov. 1985.
Dowell, "The Computerwatch Data Reduction Tool," AT&T Bell Laboratories, Whippany, New Jersey.
Fox et al., "A Neural Network Approach Towards Intrusion Detection," Harris Corporation, Government Information Systems Division, Melbourne, FL, Jul. 2, 1990.
Garvey et al., "Model-Based Intrusion Detection," Proceedings of the 14th National Computer Security Conference, Washington, DC, Oct. 1991.
Ilgun et al., State Transition Analysis: A Rule-Based Intrusion Detection Approach, IEEE Transactions on Software Engineering, vol. 21, No. 3, Mar. 1995.
Javitz et al., "The SRI IDES Statistical Anomaly Detector," Proceedings, 1991 IEEE Symposium on Security and Privacy, Oakland, California, May 1991.
Liepins et al., "Anomaly Detection: Purpose and Framework," US DOE Office of Safeguards and Security.
Lunt et al., "An Expert System to Classify and Sanitize Text," SRI International, Computer Science Laboratory, Menlo Park, CA.

(List continued on next page.)

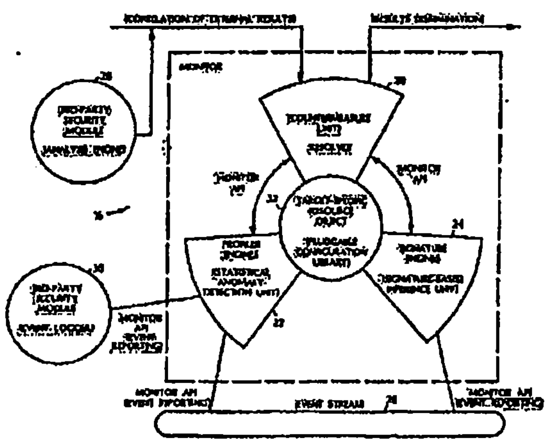
Primary Examiner—Thomas M. Heckler
(74) *Attorney, Agent, or Firm*—Fish & Richardson P.C.

(57) **ABSTRACT**

A method of network surveillance includes receiving network packets handled by a network entity and building at least one long-term and a least one short-term statistical profile from a measure of the network packets that monitors data transfers, errors, or network connections. A comparison of the statistical profiles is used to determine whether the difference between the statistical profiles indicates suspicious network activity.

27 Claims, 5 Drawing Sheets

Microfiche Appendix Included
(10 Microfiche, 956 Pages)



OTHER PUBLICATIONS

Lunt, "A Survey of Intrusion Detection Techniques," Computers & Security, 12 (1993) 405–418.

Lunt, "Automated Audit Trail Analysis and Intrusion Detection: A Survey," Proceedings of the 11th National Computer Security Conference, Baltimore, MD, Oct. 1988.

Lunt et al, "Knowledge-Based Intrusion Detection".

Lunt et al., "A Prototype Real-Time Intrusion-Detection Expert System," Proceedings of the 1988 IEEE Symposium on Security and Privacy, Apr. 1988.

Porras et al., EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances, 20th NISSC—Oct. 9, 1997.

Porras et al., Penetration State Transition Analysis A Rule-Based Intrusion Detection Approach, © 1992 IEEE.

Sebring et al., Expert Systems in Intrusion Detection: A Case Study.

Shieh et al., A Pattern-Oriented Intrusion-Detection Model and Its Applications © 1991 IEEE.

Smaha, "Haystack: An Intrusion Detection System," © 1988 IEEE Computer Society Press: Proceedings of the Fourth Aerospace Computer Security Applications Conference, 1988, pp. 37–44.

Snapp, "Signature Analysis and Communication Issues in a Distributed Intrusion Detection System," Thesis 1991.

Snapp et al., "DIDS (Distributed Intrusion Detection System)—Motivation, Architecture, and An Early Protoype, "Computer Security Laboratory, Division of Computer Science, Univ. of California, Davis, Davis, CA.

Tener, "AI & 4GL: Automated Detection and Investigation Tools," Computer Security in the Age of Information, Proceedings of the Fifth IFIP International Conference on Computer Security, W.J. Caelli (ed.).

Teng et al., "Adaptive Real-Time Anomaly Detection Using Inductively Generated Sequential Patterns," © 1990.

Vaccaro et al., "Detection of Anomalous Computer Session Activity," © 1989 IEEE.

Weiss, "Analysis of Audit and Protocol Data using Methods from Artificial Intelligence," Siemens AG, Munich, West Germany.

Winkler, "A UNIX Prototype for Intrusion and Anomaly Detection in Secure Networks," © Planning Research Corp. 1990.

Jarvis et al., The NIDES Statistical Component Description and Justification, SRI International Annual Report A010, Mar. 7, 1994.

Debar, et al., "Towards a Taxonomy of Intrusion-Detection Systems," Computers Networks 31 (1999), 805–822.

Garvey, et al., "An Inference Technique for Integrating Knowledge from Disparate Sources," Proc. IJCAI, Vancouver, B.C., Aug., 1981, 319–325.

Kaven, "The Digital Doorman," PC Magazine, Nov. 16, 1999.

Lindqvist, et al., "Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST)," Oct. 25, 1998.

* cited by examiner

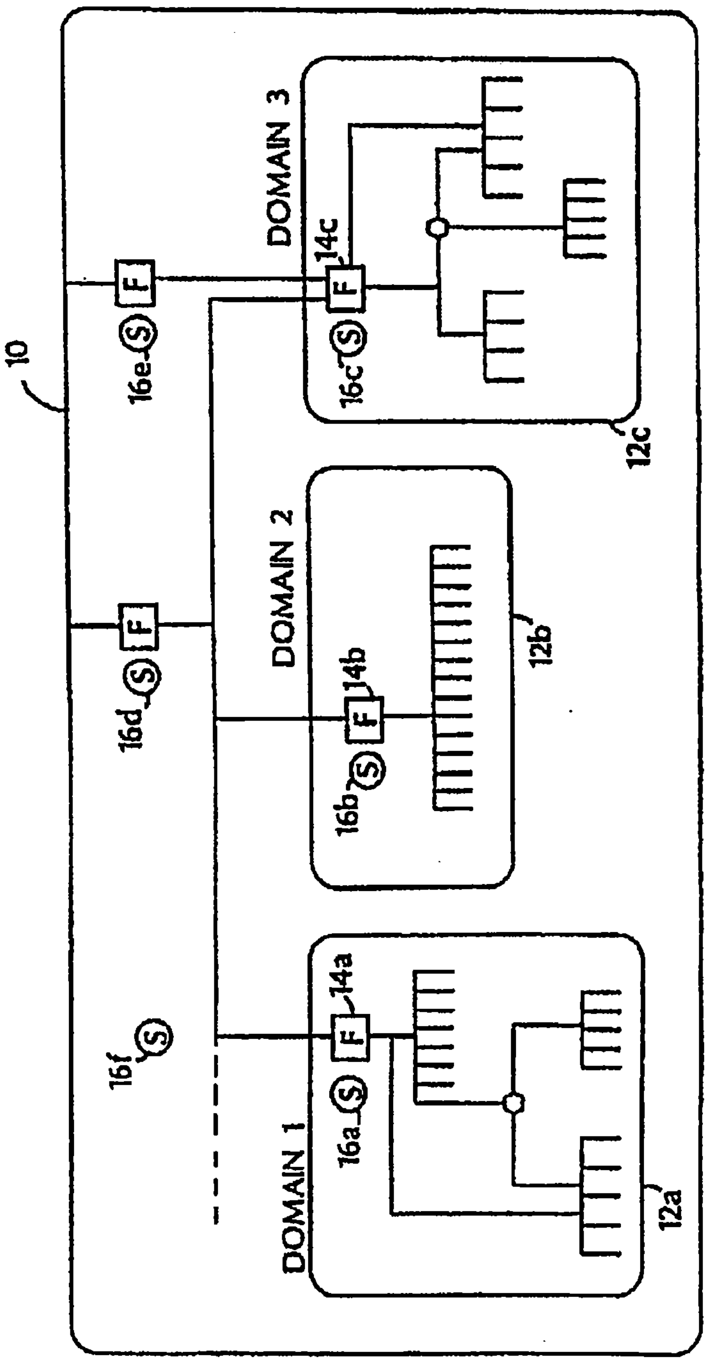

10
DOMAIN 3
DOMAIN 2
DOMAIN 1
16e
16d
16f
16c
16b
16a
14c
14b
14a
12c
12b
12a
F
S

FIG. 1

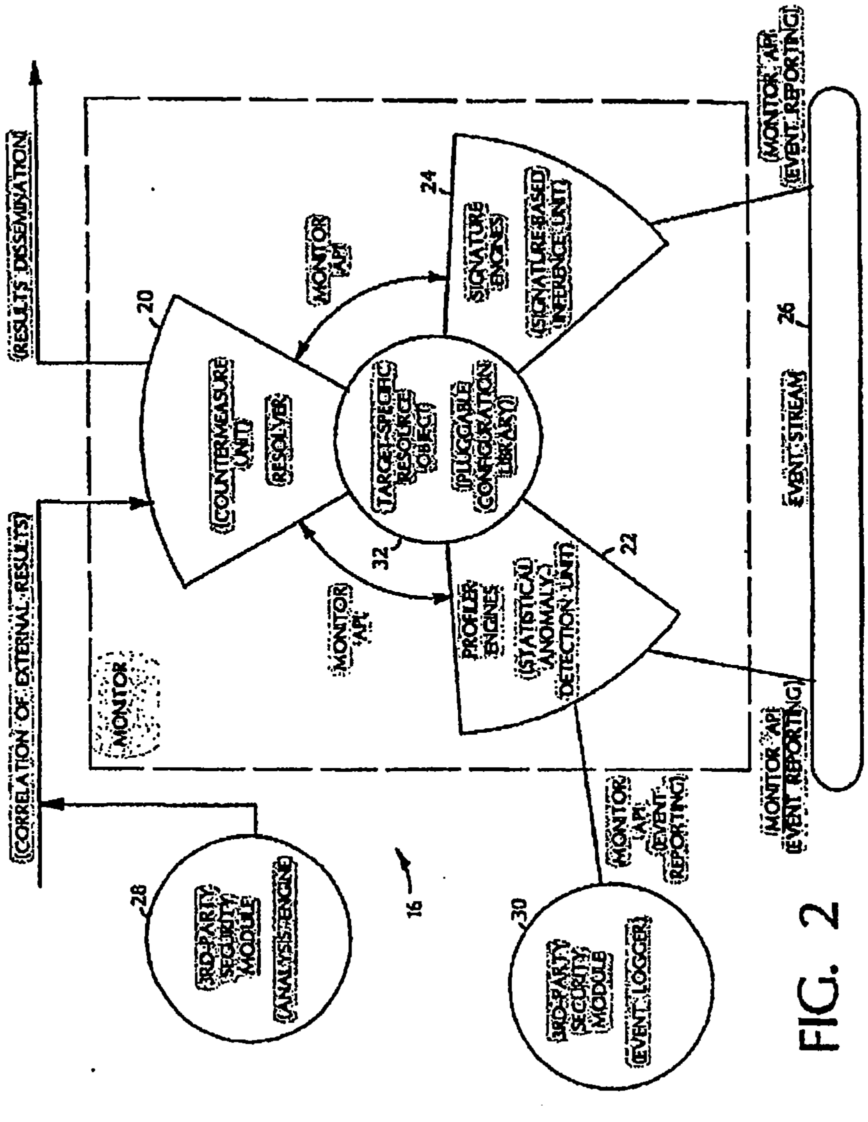

RESULTS DISSEMINATION
CORRELATION OF EXTERNAL RESULTS
MONITOR
COUNTERMEASURE UNIT
RESOLVER
TARGET-SPECIFIC RESOURCE OBJECT
PLUGGABLE CONFIGURATION LIBRARY
SIGNATURE ENGINES
SIGNATURE-BASED INFERENCE UNIT
PROFILER ENGINES
STATISTICAL ANOMALY DETECTION UNIT
MONITOR API
EVENT STREAM
MONITOR API EVENT REPORTING
3RD-PARTY SECURITY MODULE
ANALYSIS ENGINE
EVENT LOGGER
MONITOR API EVENT REPORTING
16
20
22
24
26
28
30
32

FIG. 2

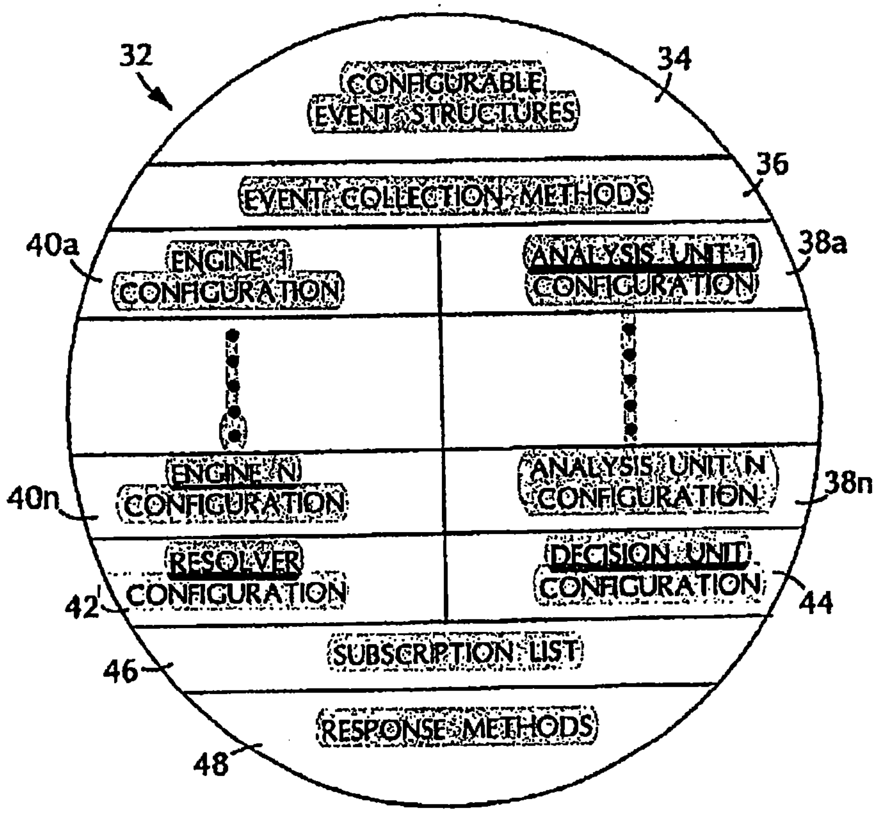

32
34
CONFIGURABLE
EVENT STRUCTURES
36
EVENT COLLECTION METHODS
40a
ENGINE 1
CONFIGURATION
38a
ANALYSIS UNIT 1
CONFIGURATION
ENGINE N
CONFIGURATION
40n
ANALYSIS UNIT N
CONFIGURATION
38n
RESOLVER
CONFIGURATION
42
DECISION UNIT
CONFIGURATION
44
46
SUBSCRIPTION LIST
RESPONSE METHODS
48

FIG. 3

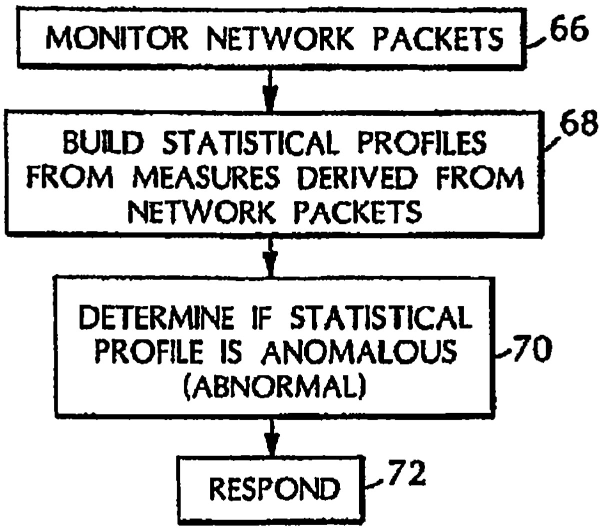

MONITOR NETWORK PACKETS
66
BUILD STATISTICAL PROFILES FROM MEASURES DERIVED FROM NETWORK PACKETS
68
DETERMINE IF STATISTICAL PROFILE IS ANOMALOUS (ABNORMAL)
70
RESPOND
72

FIG. 4

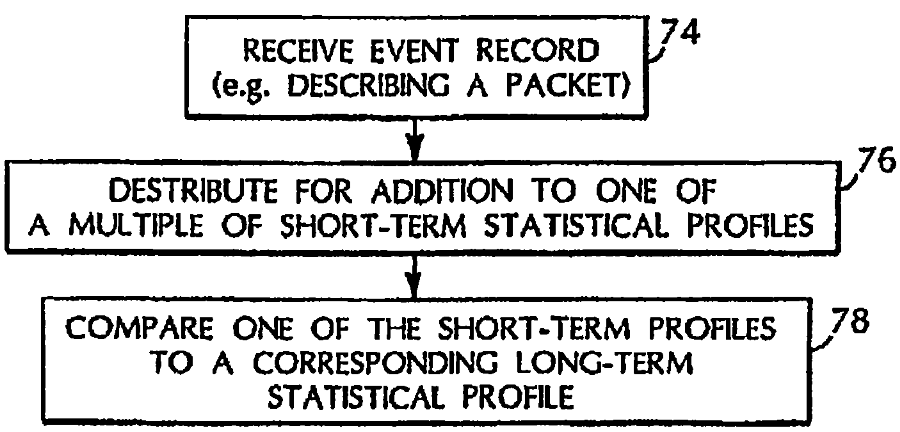
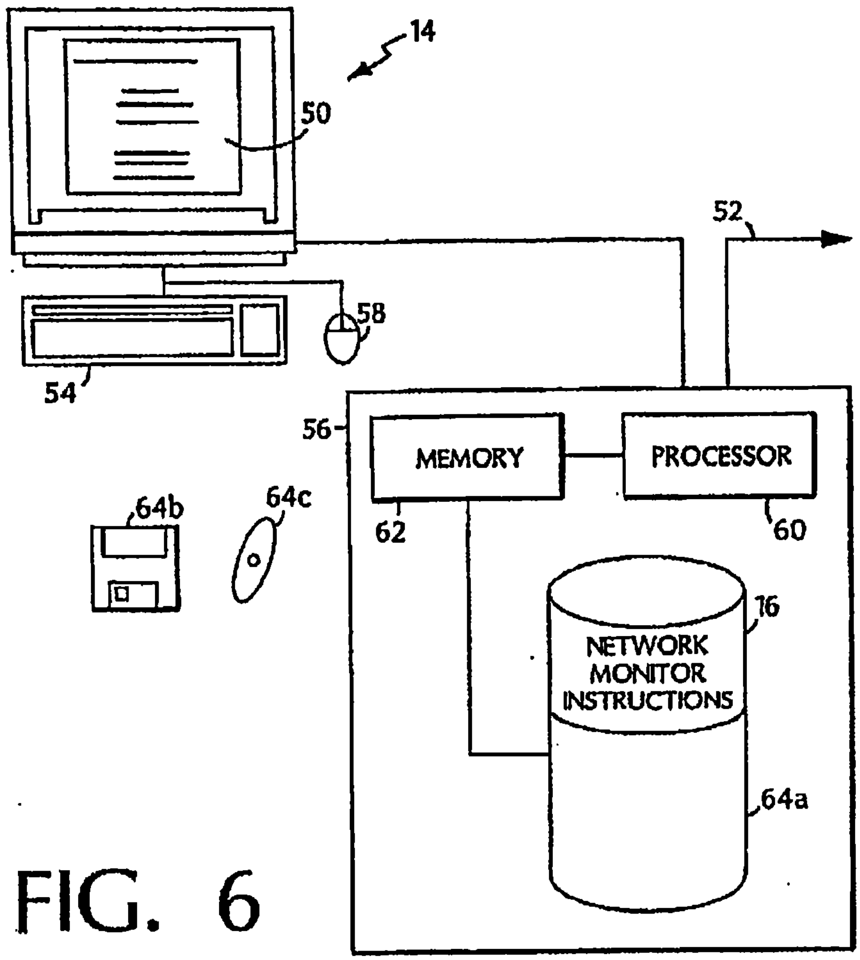

RECEIVE EVENT RECORD (e.g. DESCRIBING A PACKET)
74
DESTRIBUTE FOR ADDITION TO ONE OF A MULTIPLE OF SHORT-TERM STATISTICAL PROFILES
76
COMPARE ONE OF THE SHORT-TERM PROFILES TO A CORRESPONDING LONG-TERM STATISTICAL PROFILE
78

FIG. 5



14
50
52
58
54
56
MEMORY
PROCESSOR
62
60
64b
64c
16
NETWORK
MONITOR
INSTRUCTIONS
64a

FIG. 6

NETWORK SURVEILLANCE

REFERENCE TO GOVERNMENT FUNDING

This invention was made with Government support under Contract Number F30602-96-C-0294 awarded by DARPA. The Government has certain rights in this invention.

REFERENCE TO APPENDIX

BACKGROUND

The invention relates to computer networks.

Computer networks offer users ease and efficiency in exchanging information. 1 Networks tend to include conglomerates of integrated commercial and custom-made components, interoperating and sharing information at increasing levels of demand and capacity. Such varying networks manage a growing list of needs including transportation, commerce, energy management, communications, and defense.

2 Unfortunately, the very interoperability and sophisticated integration of technology that make networks such valuable assets also make them vulnerable to attack, and make dependence on networks a potential liability. Numerous examples of planned network attacks, such as the Internet worm, have shown how interconnectivity can be used to spread harmful program code. 3 Accidental outages such as the 1980 ARPAnet collapse and the 1990 AT&T collapse illustrate how seemingly localized triggering events can have globally disastrous effects on widely distributed systems. In addition, organized groups have performed malicious and coordinated attacks against various online targets.

SUMMARY

In general, in one aspect, a method of network surveillance includes receiving network packets (e.g., TCP/IP packets) handled by a network entity and building at least one long-term and at least one short-term statistical profile from at least one measure of the network packets that monitors data transfers, errors, or network connections. A comparison of at least one long-term and at least one short-term statistical profile is used to determine whether the difference between the short-term statistical profile and the long-term statistical profile indicates suspicious network activity.

Embodiments may include one or more of the following features. The measure may monitor data transfers by monitoring network packet data transfer commands, data transfer errors, and/or monitoring network packet data transfer volume. The measure may monitor network connections by monitoring network connection requests, network connection denials, and/or a correlation of network connections requests and network connection denials. The measure may monitor errors by monitoring error codes included in a network packet such as privilege error codes and/or error codes indicating a reason a packet was rejected.

The method may also include responding based on the determining whether the difference between a short-term statistical profile and a long-term statistical profile indicates suspicious network activity. A response may include altering analysis of network packets and/or severing a communication channel. A response may include transmitting an event record to a network monitor, such as hierarchically higher network monitor and/or a network monitor that receives event records from multiple network monitors.

The network entity may be a gateway, a router, or a proxy server. The network entity may instead be a virtual private network entity (e.g., node).

In general, in another aspect, a method of network surveillance includes monitoring network packets handled by a network entity and building a long-term and multiple short-term statistical profiles of the network packets. A comparison of one of the multiple short-term statistical profiles with the long-term statistical profile is used to determine whether the difference between the short-term statistical profiles and the long-term statistical profile indicates suspicious network activity.

Embodiments may include one or more of the following. The multiple short-term statistical profiles may monitor different anonymous FTP sessions. Building multiple short-term statistical profiles may include deinterleaving packets to identify a short-term statistical profile.

In general, in another aspect, a computer program product, disposed on a computer readable medium, includes instructions for causing a processor to receive network packets handled by a network entity and to build at least one long-term and at least one short-term statistical profile from at least one measure of the network packets that monitors data transfers, errors, or network connections. The instructions compare a short-term and a long-term statistical profile to determine whether the difference between the short-term statistical profile and the long-term statistical profile indicates suspicious network activity.

In general, in another aspect, a method of network surveillance includes receiving packets at a virtual private network entity and statistically analyzing the received packets to determine whether the packets indicate suspicious network activity. The packets may or may not be decrypted before statistical analysis.

Advantages may include one or more of the following. Using long-term and a short-term statistical profiles from measures that monitor data transfers, errors, or network connections protects network components from intrusion. As long-term profiles represent "normal" activity, abnormal activity may be detected without requiring an administrator to catalog each possible attack upon a network. Additionally, the ability to deinterleave packets to create multiple short-term profiles for comparison against a long-term profile enables the system to detect abnormal behavior that may be statistically ameliorated if only a single short-term profile was created.

The scheme of communication network monitors also protects networks from more global attacks. For example, an attack made upon one network entity may cause other entities to be alerted. Further, a monitor that collects event reports from different monitors may correlate activity to identify attacks causing disturbances in more than one network entity.

Additionally, statistical analysis of packets handled by a virtual private network enable detection of suspicious network activity despite virtual private network security techniques such as encryption of the network packets.

Other features and advantages will become apparent from the following description, including the drawings, and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram of network monitors deployed in an enterprise.

FIG. 2 is a diagram of a network monitor that monitors an event stream.

FIG. 3 is a diagram of a resource object that configures the network monitor of FIG. 2.

FIG. 4 is a flowchart illustrating network surveillance.

FIG. 5 is a flowchart illustrating multiple short-term statistical profiles for comparison against a single long-term statistical profile.

FIG. 6 is a diagram of a computer platform suitable for deployment of a network monitor.

DETAILED DESCRIPTION

Referring to FIG. 1, an enterprise 10 includes different domains 12a–12c. Each domain 12a–12c includes one or more computers offering local and network services that provide an interface for requests internal and external to the domain 12a–12c. 4 Network services include features common to many network operating systems such as mail, HTTP, FTP, remote login, network file systems, finger, Kerberos, and SNMP. Some domains 12a–12c may share trust relationships with other domains (either peer-to-peer or hierarchical). Alternatively, domains 12a–12c may operate in complete mistrust of all others, providing outgoing connections only or severely restricting incoming connections. Users may be local to a single domain or may possess accounts on multiple domains that allow them to freely establish connections throughout the enterprise 10.

5 As shown, the enterprise 10 includes dynamically deployed network monitors 16a–16f that analyze and respond to network activity and can interoperate to form an analysis hierarchy. The analysis hierarchy provides a framework for the recognition of more global threats to interdomain connectivity, including coordinated attempts to infiltrate or destroy connectivity across an entire network enterprise 10. The hierarchy includes service monitors 16a–16c, domain monitors 16d–16e, and enterprise monitors 16f.

6 Service monitors 16a–16c provide local real-time analysis of network packets (e.g., TCP/IP packets) handled by a network entity 14a–14c. Network entities include gateways, routers, firewalls, or proxy servers. A network entity may also be part of a virtual private network. A virtual private network (VPN) is constructed by using public wires to connect nodes. For example, a network could use the Internet as the medium for transporting data and use encryption and other security mechanisms to ensure that only authorized users access the network and that the data cannot be intercepted. A monitor 16a–16f can analyze packets both before and after decryption by a node of the virtual private network.

7 Information gathered by a service monitor 16a–16c can be disseminated to other monitors 16a–16f, for example, via a subscription-based communication scheme. In a subscription-based scheme client monitors subscribe to receive analysis reports produced by server monitors. As a monitor 16a–16f produces analysis reports, the monitor 16a–16f disseminates these reports asynchronously to subscribers. Through subscription, monitors 16a–16f distributed throughout a large network are able to efficiently disseminate reports of malicious activity without requiring the overhead of synchronous polling.

8 Domain monitors 16d–16e perform surveillance over all or part of a domain 12a–12c. Domain monitors 16d–16e

correlate intrusion reports disseminated by individual service monitors 16a–16c, providing a domain-wide perspective of activity (or patterns of activity). In addition to domain surveillance, domain monitors 16a–16c can reconfigure system parameters, interface with other monitors beyond a domain, and report threats against a domain 12a–12c to administrators. Domain monitors 16d–16e can subscribe to service monitors 16a–16c. Where mutual trust among domains 12a–12c exists, domain monitors 16d–16e may establish peer relationships with one another. Peer-to-peer subscription allows domain monitors 16d–16e to share analysis reports produced in other domains 12a–12c. Domain monitors 16d–16e may use such reports to dynamically sensitize their local service monitors 16a–16c to malicious activity found to be occurring outside a domain 12a–12c. Domain monitors 16d–16e may also operate within an enterprise hierarchy where they disseminate analysis reports to enterprise monitors 16f for global correlation. 9

Enterprise monitors 16f correlate activity reports produced across the set of monitored domains 12a–12c. 10 Enterprise 10 surveillance may be used where domains 12a–12c are interconnected under the control of a single organization, such as a large privately owned WAN (Wide Area Network). 11 The enterprise 10, however, need not be stable in its configuration or centrally administered. For example, the enterprise 10 may exist as an emergent entity through new interconnections of domains 12a–12c. 12 Enterprise 10 surveillance is very similar to domain 12a–12c surveillance: an enterprise monitor 16f subscribes to various domain monitors 16d–16e, just as the domain monitors 16d–16e subscribed to various service monitors 16a–16c. The enterprise monitor 16f (or monitors, as it would be important to avoid centralizing any analysis) focuses on network-wide threats such as Internet worm-like attacks, attacks repeated against common network services across domains, or coordinated attacks from multiple domains against a single domain. As an enterprise monitor 16f recognizes commonalities in intrusion reports across domains (e.g., the spreading of a worm or a mail system attack repeated throughout the enterprise 10), the monitor 16f can help domains 12a–12c counter the attack and can sensitize other domains 12a–12c to such attacks before they are affected. 13 Through correlation and sharing of analysis reports, reports of problems found by one monitor 16a–16f may propagate to other monitors 16a–16f throughout the network. Interdomain event analysis is vital to addressing more global, information attacks against the entire enterprise 10.

Referring to FIG. 2, each monitor 16 includes one or more analysis engines 22, 24. These engines 22, 24 can be dynamically added, deleted, and modified as necessary. In the dual-analysis configuration shown, a monitor 16 instantiation includes a signature analysis engine 22 and a statistical profiling engine 24. In general, a monitor 16 may include additional analysis engines that may implement other forms of analysis. A monitor 16 also includes a resolver 20 that implements a response policy and a resource object 32 that configures the monitor 16. The monitors 16 incorporate an application programmers' interface (API) that enhances encapsulation of monitor functions and eases integration of third-party intrusion-detection tools 28, 30. 14

Each monitor 16 can analyze event records that form an event stream. The event stream may be derived from a variety of sources such as TCP/IP network packet contents or event records containing analysis reports disseminated by other monitors. 15 For example, an event record can be formed from data included in the header and data segment of a network packet. The volume of packets transmitted and

received, however, dictates careful assessment of ways to select and organize network packet information into event record streams.

Selection of packets can be based on different criteria. Streams of event records can be derived from discarded traffic (i.e., packets not allowed through the gateway because they violate filtering rules), pass-through traffic (i.e., packets allowed into the internal network from external sources), packets having a common protocol (e.g., all ICMP (Internet Control Message Protocol) packets that reach the gateway), packets involving network connection management (e.g., SYN, RESET, ACK, [window resize]), and packets targeting ports to which an administrator has not assigned any network service and that also remain unblocked by the firewall. Event streams may also be based on packet source addresses (e.g., packets whose source addresses match well-known external sites such as satellite offices or have raised suspicion from other monitoring efforts) or destination addresses (e.g., packets whose destination addresses match a given internal host or workstation). Selection can also implement application-layer monitoring (e.g., packets targeting a particular network service or application). Event records can also be produced from other sources of network packet information such as report logs produced by network entities. Event streams can be of very fine granularity. For example, a different stream might be derived for commands received from different commercial web-browsers since each web-browser produces different characteristic network activity.

A monitor 16 can also construct interval summary event records, which contain accumulated network traffic statistics (e.g., number of packets and number of kilobytes transferred). These event records are constructed at the end
16 of each interval (e.g., once per N seconds). Event records are forwarded to the analysis engines 22, 24 for analysis.

The profile engine 22 can use a wide range of multivariate statistical measures to profile network activity indicated by an event stream. A statistical score represents how closely currently observed usage corresponds to the established
17 patterns of usage. The profiler engine 22 separates profile management and the mathematical algorithms used to assess the anomaly of events. The profile engine 22 may use a statistical analysis technique described in A. Valdes and D. Anderson, "Statistical Methods for Computer Usage Anomaly Detection Using NIDES", Proceedings of the Third International Workshop on Rough Sets and Soft Computing, January 1995, which is incorporated by reference in its entirety. Such an engine 22 can profile network activity via one or more variables called measures. Measures can be categorized into four classes: categorical, continuous, intensity, and event distribution measures.

Categorical measures assume values from a discrete, nonordered set of possibilities. Examples of categorical measures include network source and destination addresses, commands (e.g., commands that control data transfer and manage network connections), protocols, error codes (e.g., privilege violations, malformed service requests, and malformed packet codes), and port identifiers. The profiler engine 22 can build empirical distributions of the category values encountered, even if the list of possible values is open-ended. The engine 22 can have mechanisms for "aging out" categories whose long-term probabilities drop below a threshold.

Continuous measures assume values from a continuous or ordinal set. Examples include inter-event time (e.g., difference in time stamps between consecutive events from the same stream), counting measures such as the number of errors of a particular type observed in the recent past, the volume of data transfers over a period of time, and network traffic measures (number of packets and number of kilobytes). The profiler engine 22 treats continuous measures by first allocating bins appropriate to the range of values of the underlying measure, and then tracking the frequency of observation of each value range. In this way, multi-modal distributions are accommodated and much of the computational machinery used for categorical measures is shared. Continuous measures are useful not only for intrusion detection, but also to support the monitoring of the health and status of the network from the perspective of connectivity and throughput. For example, a measure of traffic volume maintained can detect an abnormal loss in the data rate of received packets when this volume falls outside historical norms. This sudden drop can be specific both to the network entity being monitored and to the time of day (e.g., the average sustained traffic rate for a major network artery is much different at 11:00 a.m. than at midnight).

Intensity measures reflect the intensity of the event stream (e.g., number of ICMP packets) over specified time intervals (e.g., 1 minute, 10 minutes, and 1 hour). Intensity measures are particularly suited for detecting flooding attacks, while also providing insight into other anomalies.

Event distribution measures are meta-measures that describes how other measures in the profile are affected by each event. For example, an "ls" command in an FTP session affects the directory measure, but does not affect measures related to file transfer. This measure is not interesting for all event streams. For example, all network-traffic event records affect the same measures (number of packets and kilobytes) defined for that event stream, so the event distribution does not change. On the other hand, event distribution measures are useful in correlative analysis performed by a monitor 16*f* that receives reports from other monitors 16*a*–16*f*.

The system maintains and updates a description of behavior with respect to these measure types in an updated profile. The profile is subdivided into short-term and long-term profiles. The short-term profile accumulates values between updates, and exponentially ages (e.g., weighs data based on how long ago the data was collected) values for comparison to the long-term profile. As a consequence of the aging mechanism, the short-term profile characterizes recent activity, where "recent" is determined by a dynamically configurable aging parameters. At update time (typically, a time of low system activity), the update function folds the short-term values observed since the last update into the long-term profile, and the short-term profile is cleared. The long-term profile is itself slowly aged to adapt to changes in subject activity. Anomaly scoring compares related attributes in the short-term profile against the long-term profile. As all evaluations are done against empirical distributions, no assumptions of parametric distributions are made, and multi-modal and categorical distributions are accommodated. Furthermore, the algorithms require no a priori knowledge of intrusive or exceptional activity.

The statistical algorithm adjusts a short-term profile for the measure values observed in the event record. The distribution of recently observed values is compared against the long-term profile, and a distance between the two is obtained. The difference is compared to a historically adaptive deviation. The empirical distribution of this deviation is transformed to obtain a score for the event. Anomalous events are those whose scores exceed a historically adaptive score threshold based on the empirical score distribution.

This nonparametric approach handles all measure types and makes no assumptions on the modality of the distribution for continuous measures.

[18] Profiles are provided to the computational engine as classes defined in the resource object 32. The mathematical functions for anomaly scoring, profile maintenance, and updating do not require knowledge of the data being analyzed beyond what is encoded in the profile class. Event collection interoperability supports translation of the event stream to the profile and measure classes. At that point, analysis for different types of monitored entities is mathematically similar. This approach imparts great flexibility to the analysis in that fading memory constants, update frequency, measure type, and so on are tailored to the network entity being monitored.

The measure types described above can be used individually or in combination to detect network packet attributes characteristic of intrusion. Such characteristics include large data transfers (e.g., moving or downloading files), an increase in errors (e.g., an increase in privilege violations or network packet rejections), network connection activity, and abnormal changes in network volume.

As shown, the monitor 16 also includes a signature engine 24. [19] The signature engine 24 maps an event stream against abstract representations of event sequences that are known to indicate undesirable activity. [47] Signature-analysis objectives depend on which layer in the hierarchical analysis scheme the signature engine operates. Service monitor 16a–16c signature engines 24 attempt to monitor for attempts to penetrate or interfere with the domain's operation. [20] The signature engine scans the event stream for events that represent attempted exploitations of known attacks against the service, or other activity that stands alone as warranting a response from the monitor. Above the service layer, signature engines 24 scan the aggregate of intrusion reports from service monitors in an attempt to detect more global coordinated attack scenarios or scenarios that exploit interdependencies among network services. Layering signature engine analysis enables the engines 24 to avoid misguided searches along incorrect signature paths in addition to distributing the signature analysis.

A signature engines 24 can detect, for example, address spoofing, tunneling, source routing, SATAN attacks, and abuse of ICMP messages ("Redirect" and "Destination Unreachable" messages in particular). Threshold analysis is a rudimentary, inexpensive signature analysis technique that records the occurrence of specific events and, as the name implies, detects when the number of occurrences of that event surpasses a reasonable count. For example, monitors can encode thresholds to monitor activity such as the number of fingers, pings, or failed login requests to accounts such as guest, demo, visitor, anonymous FTP, or employees who have departed the company.

Signature engine 24 can also examine the data portion of packets in search of a variety of transactions that indicate suspicious, if not malicious, intentions by an external client. The signature engine 24, for example, can parse FTP traffic traveling through the firewall or router for unwanted transfers of configuration or specific system data, or anonymous requests to access non-public portions of the directory structure. Similarly, a monitor can analyze anonymous FTP sessions to ensure that the file retrievals and uploads/modifications are limited to specific directories. Additionally, signature analysis capability can extend to session analyses of complex and dangerous, but highly useful, services like HTTP or Gopher.

Signature analysis can also scan traffic directed at unused ports (i.e., ports to which the administrator has not assigned a network service). Here, packet parsing can be used to study network traffic after some threshold volume of traffic, directed at an unused port, has been exceeded. A signature engine 24 can also employ a knowledge base of known telltale packets that are indicative of well-known network-service protocol traffic (e.g., FTP, Telnet, SMTP, HTTP). The signature engine 24 then determines whether the unknown port traffic matches any known packet sets. Such comparisons could lead to the discovery of network services that have been installed without an administrator's knowledge.

The analysis engines 22, 24 receive large volumes of events and produce smaller volumes of intrusion or suspicion reports that are then fed to the resolver 20. [21] [22] The resolver 20 is an expert system that receives the intrusion and suspicion reports produced by the analysis engines 22, 24 and reports produced externally by other analysis engines to which it subscribes. Based on these reports, the resolver 20 invokes responses. Because the volume of intrusion and suspicion reports is lower than the volume of events received by the analysis engines 22, 24, the resolver 20 can afford the more sophisticated demands of configuration maintenance and managing the response handling and external interfaces necessary for monitor operation. Furthermore, the resolver 20 adds to extensibility by providing the subscription interface through which third-party analysis tools 28, 30 can interact and participate in the hierarchical analysis scheme.

Upon its initialization, the resolver 20 initiates authentication and subscription sessions with those monitors 16a–16f whose identities appear in the monitor's 16 subscription-list (46 FIG. 3). The resolver 20 also handles all incoming requests by subscribers, which must authenticate themselves to the resolver 20. Once a subscription session is established with a subscriber monitor, the resolver 20 acts as the primary interface through which configuration requests are received and intrusion reports are disseminated. [23]

Thus, resolvers 20 can request and receive reports from other resolvers at lower layers in the analysis hierarchy. The resolver 20 forwards analysis reports received from subscribees to the analysis engines 22, 24. This tiered collection and correlation of analysis results allows monitors 16a–16f to represent and profile global malicious or anomalous activity that is not visible locally. [24]

In addition to external-interface responsibilities, the resolver 20 operates as a fully functional decision engine, capable of invoking real-time response measures in response to malicious or anomalous activity reports produced by the analysis engines. [25] [26] The resolver 20 also operates as the center of intramonitor communication. As the analysis engines 22, 24 build intrusion and suspicion reports, they propagate these reports to the resolver 20 for further correlation, response, and dissemination to other monitors 16a–16f. The resolver 20 can also submit runtime configuration requests to the analysis engines 22, 24, for example, to increase or decrease the scope of analyses (e.g., enable or disable additional signature rules) based on various operating metrics. These configuration requests could be made as a result of encountering other intrusion reports from other subscribers. For example, a report produced by a service monitor 16a–16c in one domain could be propagated to an enterprise monitor 16f, which in turn sensitizes service monitors in other domains to the same activity.

The resolver 20 also operates as the interface mechanism between administrators and the monitor 16. From the per- [27]

[27] spective of a resolver 20, the administrator interface is simply a subscribing service to which the resolver 20 may submit reports and receive configuration requests. An administrative interface tool can dynamically subscribe and unsubscribe to any of the deployed resolvers 20, as well as submit configuration requests and asynchronous probes as desired.

[28] The monitors 16a–16f incorporate a bidirectional messaging system that uses a standard interface specification for communication within and between monitor elements and external modules. Using this interface specification, third-party modules 28, 30 can communicate with monitors. For example, third-party modules 28 can submit event records to the analysis engines 22, 24 for processing. Additionally, third-party modules 30 may also submit and receive analysis results via the resolver's 20 external interfaces. Thus, third-party modules 28, 30 can incorporate the results from monitors into other surveillance efforts or contribute their results to other monitors 16a–16f. [29] Lastly, the monitor's 16 internal API allows third-party analysis engines to be linked directly into the monitor boundary.

[30] The message system operates under an asynchronous communication model for handling results dissemination and processing that is generically referred to as subscription-based message passing. Component interoperation is client/server-based, where a client module may subscribe to receive event data or analysis results from servers. Once a subscription request is accepted by the server, the server module forwards events or analysis results to the client automatically as data becomes available, and may dynamically reconfigure itself as requested by the client's control requests. This asynchronous model reduces the need for client probes and acknowledgments.

[31] The interface supports an implementation-neutral communication framework that separates the programmer's interface specification and the issues of message transport. The interface specification embodies no assumptions about implementation languages, host platform, or a network. The transport layer is architecturally isolated from the internals of the monitors so that transport modules may be readily introduced and replaced as protocols and security requirements are negotiated between module developers. [32] The interface specification involves the definition of the messages that the various intrusion-detection modules must convey to one another and how these messages should be processed. The message structure and content are specified in a completely implementation-neutral context.

[33] Both intramonitor and intermonitor communication employ identical subscription-based client-server models. With respect to intramonitor communication, the resolver 20 operates as a client to the analysis engines, and the analysis engines 22, 24 operate as clients to the event filters. Through the internal message system, the resolver 20 submits configuration requests to the analysis engines 22, 24, and receives from the analysis engines 22, 24 their analysis results. The analysis engines 22, 24 operate as servers providing the resolver 20 with intrusion or suspicion reports either asynchronously or upon request. Similarly, the analysis engines 22, 24 are responsible for establishing and maintaining a communication link with an event collection method (or event filter) and prompting the reconfiguration of the collection method's filtering semantics when necessary.

[34] Intermonitor communication also operates using the subscription-based hierarchy. A domain monitor 16d–16e subscribes to the analysis results produced by service monitors 16a–16c, and then propagates its own analytical reports

to its parent enterprise monitor 16f. The enterprise monitor 16f operates as a client to one or more domain monitors 16d–16e, allowing them to correlate and model enterprise-wide activity from the domain-layer results. Domain monitors 16d–16e operate as servers to the enterprise monitors 16f, and as clients to the service monitors 16a–16c deployed throughout their domains 12a–12c. This message scheme can operate substantially the same if correlation were to continue at higher layers of abstraction beyond enterprise 10 analysis. [34]

Intramonitor and intermonitor programming interfaces are substantially the same. These interfaces can be subdivided into five categories of interoperation: channel initialization and termination, channel synchronization, dynamic configuration, server probing, and report/event dissemination. Clients are responsible for initiating and terminating channel sessions with servers. Clients are also responsible for managing channel synchronization in the event of errors in message sequencing or periods of failed or slow response (i.e., "I'm alive" confirmations). Clients may also submit dynamic configuration requests to servers. For example, an analysis engine 22, 24 may request an event collection method to modify its filtering semantics. Clients may also probe servers for report summaries or additional event information. Lastly, servers may send clients intrusion/suspicion reports in response to client probes or in an asynchronous dissemination mode. [35]

The second part of the message system framework involves specification of a transport mechanism used to establish a given communication channel between monitors 16a–16f or possibly between a monitor 16a–16f and a third-party security module. All implementation dependencies within the message system framework are addressed by pluggable transport modules. Transport modules are specific to the participating intrusion-detection modules, their respective hosts, and potentially to the network—should the modules require cross-platform interoperation. Instantiating a monitor 16a–16f may involve incorporation of the necessary transport module(s) (for both internal and external communication). [36]

The transport modules that handle intramonitor communication may be different from the transport modules that handle intermonitor communication. This allows the intramonitor transport modules to address security and reliability issues differently than how the intermonitor transport modules address security and reliability. While intramonitor communication may more commonly involve interprocess communication within a single host, intermonitor communication will most commonly involve cross-platform networked interoperation. For example, the intramonitor transport mechanisms may employ unnamed pipes which provides a kernel-enforced private interprocess communication channel between the monitor 16 components (this assumes a process hierarchy within the monitor 16 architecture). The monitor's 16 external transport, however, will more likely export data through untrusted network connections and thus require more extensive security management. To ensure the security and integrity of the message exchange, the external transport may employ public/private key authentication protocols and session key exchange. Using this same interface, third-party analysis tools may authenticate and exchange analysis results and configuration information in a well-defined, secure manner. [37]

The pluggable transport permits flexibility in negotiating security features and protocol usage with third parties. Incorporation of a commercially available network management system can deliver monitoring results relating to security, reliability, availability, performance, and other

attributes. The network management system may in turn subscribe to monitor produced results in order to influence network reconfiguration.

[38] All monitors (service, domain, and enterprise) 16a–16f use the same monitor code-base. However, monitors may include different resource objects 32 having different configuration data and methods. This reusable software architecture can reduce implementation and maintenance efforts. Customizing and dynamically configuring a monitor 16 thus becomes a question of building and/or modifying the resource object 32.

[39] Referring to FIG. 3, the resource object 32 contains the operating parameters for each of the monitor's 16 components as well as the analysis semantics (e.g., the profiler engine's 22 measure and category definition, or the signature engine's 24 penetration rule-base) necessary to process an event stream. After defining a resource object 32 to implement a particular set of analyses on an event stream, the resource object 32 may be reused by other monitors 16 deployed to analyze equivalent event streams. For example, the resource object 32 for a domain's router may be reused as other monitors 16 are deployed for other routers in a domain 12a–12c. A library of resource objects 32 provides prefabricated resource objects 32 for commonly available network entities.

[40] The resource object 32 provides a pluggable configuration module for tuning the generic monitor code-base to a specific event stream. The resource object 32 includes configurable event structures 34, analysis unit configuration 38a–38n, engine configuration 40a–40n, resolver configuration 42, decision unit configuration 44, subscription list data 46, and response methods 48.

[41] Configurable event structures 34 define the structure of event records and analysis result records. The monitor code-base maintains no internal dependence on the content or format of any given event stream or the analysis results produced from analyzing the event stream. Rather, the resource object 32 provides a universally applicable syntax for specifying the structure of event records and analysis results. Event records are defined based on the contents of an event stream(s). Analysis result structures are used to package the findings produced by analysis engines. Event records and analysis results are defined similarly to allow the eventual hierarchical processing of analysis results as event records by subscriber monitors.

Event-collection methods 36 gather and parse event records for analysis engine processing. Processing by analysis engines is controlled by engine configuration 40a–40n variables and data structures that specify the operating configuration of a fielded monitor's analysis engine(s). The resource object 32 maintains a separate collection of operating parameters for each analysis engine instantiated in the monitor 16. Analysis unit configuration 38a–38n include configuration variables that define the semantics employed by the analysis engine to process the event stream.

[42] The resolver configuration 42 includes operating parameters that specify the configuration of the resolver's internal modules. [43] The decision unit configuration 44 describes semantics used by the resolver's decision unit for merging the analysis results from the various analysis engines. The semantics include the response criteria used to invoke countermeasure handlers. A resource object 32 may also include response methods 48. Response methods 48 include preprogrammed countermeasure methods that the resolver may invoke as event records are received. A response method 48 includes evaluation metrics for determining the circumstances under which the method should be invoked. These metrics include a threshold metric that corresponds to the measure values and scores produced by the profiler engine 22 and severity metrics that correspond to subsets of the associated attack sequences defined within the resource object 32.

Countermeasures range from very passive responses, such as report dissemination to other monitors 16a–16f or administrators, to highly aggressive actions, such as severing a communication channel or the reconfiguration of logging facilities within network components (e.g., routers, firewalls, network services, audit daemons). An active response may invoke handlers that validate the integrity of network services or other assets to ensure that privileged network services have not been subverted. Monitors 16a–16f may invoke probes in an attempt to gather as much counterintelligence about the source of suspicious traffic by using features such as traceroute or finger.

[44] The resource object 32 may include a subscription list 46 that includes information necessary for establishing subscription-based communication sessions, which may include network address information and public keys used by the monitor to authenticate potential clients and servers. The subscription list 46 enables transmission or reception of messages that report malicious or anomalous activity between monitors. [45] The most obvious examples where relationships are important involve interdependencies among network services that make local policy decisions. For example, the interdependencies between access checks performed during network file system mounting and the IP mapping of the DNS service. An unexpected mount monitored by the network file system service may be responded to differently if the DNS monitor informs the network file system monitor of suspicious updates to the mount requestor's DNS mapping.

[46] The contents of the resource object 32 are defined and utilized during monitor 16 initialization. In addition, these fields may be modified by internal monitor 16 components, and by authorized external clients using the monitor's 16 API. Modifying the resource object 32 permits adaptive analysis of an event stream, however, it also introduces a potential stability problem if dynamic modifications are not tightly restricted to avoid cyclic modifications. To address this issue, monitors 16 can be configured to accept configuration requests from only higher-level monitors 16.

Referring to FIG. 4, a monitor performs network surveillance by monitoring 66 a stream of network packets. The monitor builds a statistical model of network activity from the network packets, for example, by building 68 long-term and short-term statistical profiles from measures derived from the network packets. The measures include measures that can show anomalous network activity characteristic of network intrusion such as measures that describe data transfers, network connections, privilege and network errors, and abnormal levels of network traffic. The monitor can compare 70 the long-term and short-term profiles to detect suspicious network activity. Based on this comparison, the monitor can respond 72 by reporting the activity to another monitor or by executing a countermeasure response. More information can be found in P. Porras and A. Valdes "Live Traffic Analysis of TCP/IP Gateways", Networks and Distributed Systems Security Symposium, March 1998, which is incorporated by reference in its entirety.

A few examples can illustrate this method of network surveillance. Network intrusion frequently causes large data

transfers, for example, when an intruder seeks to download sensitive files or replace system files with harmful substitutes. A statistical profile to detect anomalous data transfers might include a continuous measure of file transfer size, a categorical measure of the source or destination directory of the data transfer, and an intensity measure of commands corresponding to data transfers (e.g., commands that download data). These measures can detect a wide variety of data transfer techniques such as a large volume of small data transfers via e-mail or downloading large files en masse. The monitor may distinguish between network packets based on the time such packets were received by the network entity, permitting statistical analysis to distinguish between a normal data transfer during a workday and an abnormal data transfer on a weekend evening.

Attempted network intrusion may also produce anomalous levels of errors. For example, categorical and intensity measures derived from privilege errors may indicate attempts to access protected files, directories, or other network assets. Of course, privilege errors occur during normal network operation as users mistype commands or attempt to perform an operation unknowingly prohibited. By comparing the long-term and short-term statistical profiles, a monitor can distinguish between normal error levels and levels indicative of intrusion without burdening a network administrator with the task of arbitrarily setting an unvarying threshold. Other measures based on errors, such as codes describing why a network entity rejected a network packet enable a monitor to detect attempts to infiltrate a network with suspicious packets.

Attempted network intrusion can also be detected by measures derived from network connection information. For example, a measure may be formed from the correlation (e.g., a ratio or a difference) of the number of SYN connection request messages with the number of SYN_ACK connection acknowledgment messages and/or the number of ICMP messages sent. Generally, SYN requests received should balance with respect to the total of SYN_ACK and ICMP messages sent. That is, flow into and out-of a network entity should be conserved. An imbalance can indicate repeated unsuccessful attempts to connect with a system, perhaps corresponding to a methodical search for an entry point to a system. Alternatively, intensity measures of transport-layer connection requests, such as a volume analysis of SYN-RST messages, could indicate the occurrence of a SYN-attack against port availability or possibly port-scanning. Variants of this can include intensity measures of TCP/FIN messages, considered a more stealthy form of port scanning.

Many other measures can detect network intrusion. For example, "doorknob rattling," testing a variety of potentially valid commands to gain access (e.g., trying to access a "system" account with a password of "system"), can be detected by a variety of categorical measures. A categorical measure of commands included in network packets can identify an unusual short-term set of commands indicative of "doorknob-rattling." Similarly, a categorical measure of protocol requests may also detect an unlikely mix of such requests.

Measures of network packet volume can also help detect malicious traffic, such as traffic intended to cause service denials or perform intelligence gathering, where such traffic may not necessarily be violating filtering policies. A measure reflecting a sharp increase in the overall volume of discarded packets as well as a measure analyzing the disposition of the discarded packets can provide insight into unintentionally malformed packets resulting from poor line quality or internal errors in neighboring hosts. High volumes of discarded packets can also indicate more maliciously intended transmissions such as scanning of UPD ports or IP address scanning via ICMP echoes. Excessive number of mail expansion request commands (EXPN) may indicate intelligence gathering, for example, by spammers.

A long-term and short-term statistical profile can be generated for each event stream. Thus, different event streams can "slice" network packet data in different ways. For example, an event stream may select only network packets having a source address corresponding to a satellite office. Thus, a long-term and short-term profile will be generated for the particular satellite office. Thus, although a satellite office may have more privileges and should be expected to use more system resources than other external addresses, a profile of satellite office use can detect "address spoofing" (i.e., modifying packet information to have a source address of the satellite office).

The same network packet event may produce records in more than one event stream. For example, one event stream may monitor packets for FTP commands while another event stream monitors packets from a particular address. In this case, an FTP command from the address would produce an event record in each stream.

Referring to FIG. 5, a monitor may also "deinterleave." That is, the monitor may create and update 74, 76 more than one short-term profile for comparison 78 against a single long-term profile by identifying one of the multiple short-term profiles that will be updated by an event record in an event stream. For example, at any one time a network entity may handle several FTP "anonymous" sessions. If each network packet for all anonymous sessions were placed in a single short-term statistical profile, potentially intrusive activity of one anonymous session may be statistically ameliorated by non-intrusive sessions. By creating and updating short-term statistical profiles for each anonymous session, each anonymous session can be compared against the long-term profile of a normal FTP anonymous session. Deinterleaving can be done for a variety of sessions including HTTP sessions (e.g., a short-term profile for each browser session).

Referring to FIG. 6, a computer platform 14 suitable for executing a network monitor 16 includes a display 50, a keyboard 54, a pointing device 58 such as a mouse, and a digital computer 56. The digital computer 56 includes memory 62, a processor 60, a mass storage device 64*a*, and other customary components such as a memory bus and peripheral bus. The platform 14 may further include a network connection 52.

Mass storage device 64*a* can store instructions that form a monitor 16. The instructions may be transferred to memory 62 and processor 60 in the course of operation. The instructions 16 can cause the display 50 to display images via an interface such as a graphical user interface. Of course, instructions may be stored on a variety of mass storage devices such as a floppy disk 64*b*, CD-ROM 64*c*, or PROM (not shown).

Other embodiments are within the scope of the following claims.

What is claimed is:

1. A method of network surveillance, comprising:

receiving network packets handled by a network entity;

building at least one long-term and at least one short-term statistical profile from at least one measure of the network packets, the at least one measure monitoring data transfers, errors, or network connections;

comparing at least one long-term and at least one short-term statistical profile; and

determining whether the difference between the short-term statistical profile and the long-term statistical profile indicates suspicious network activity.

2. The method of claim 1, wherein the measure monitors data transfers by monitoring network packet data transfer commands.

3. The method of claim 1, wherein the measure monitors data transfers by monitoring network packet data transfer errors.

4. The method of claim 1, wherein the measure monitors data transfers by monitoring network packet data transfer volume.

5. The method of claim 1, wherein the measure monitors network connections by monitoring network connection requests.

6. The method of claim 1, wherein the measure monitors network connections by monitoring network connection denials.

7. The method of claim 1, wherein the measure monitors network connections by monitoring a correlation of network connections requests and network connection denials.

8. The method of claim 1, wherein the measure monitors errors by monitoring error codes included in a network packet.

9. The method of claim 8, wherein an error code comprises a privilege error code.

10. The method of claim 8, wherein an error code comprises an error code indicating a reason a packet was rejected.

11. The method of claim 1, further comprising responding based on the determining whether the difference between the short-term statistical profile and the long-term statistical profile indicates suspicious network activity.

12. The method of claim 11, wherein responding comprises transmitting an event record to a network monitor.

13. The method of claim 12, wherein transmitting the event record to a network monitor comprises transmitting the event record to a hierarchically higher network monitor.

14. The method of claim 13, wherein transmitting the event record to a network monitor comprises transmitting the event record to a network monitor that receives event records from multiple network monitors.

15. The method of claim 14, wherein the monitor that receives event records from multiple network monitors comprises a network monitor that correlates activity in the multiple network monitors based on the received event records.

16. The method of claim 11, wherein responding comprises altering analysis of the network packets.

17. The method of claim 11, wherein responding comprises severing a communication channel.

18. The method of claim 1, wherein the network packets comprise TCP/IP packets.

19. The method of claim 1, wherein the network entity comprises a gateway, a router, or a proxy server.

20. The method of claim 1, wherein the network entity comprises a virtual private network entity.

21. A method of network surveillance, comprising:

monitoring network packets handled by a network entity;

building a long-term and multiple short-term statistical profiles of the network packets;

comparing one of the multiple short-term statistical profiles with the long-term statistical profile; and

determining whether the difference between the one of the multiple short-term statistical profiles and the long-term statistical profile indicates suspicious network activity.

22. The method of claim 21, wherein the multiple short-term statistical profiles comprise profiles that monitor different anonymous FTP sessions.

23. The method of claim 21, wherein building multiple short-term statistical profiles comprises deinterleaving packets to identify a short-term statistical profile.

24. A computer program product, disposed on a computer readable medium, the product including instructions for causing a processor to:

receive network packets handled by a network entity;

build at least one long-term and at least one short-term statistical profile from at least one measure of the network packets, the measure monitoring data transfers, errors, or network connections;

compare at least one short-term and at least one long-term statistical profile; and

determine whether the difference between the short-term statistical profile and the long-term statistical profile indicates suspicious network activity.

25. A method of network surveillance, comprising:

receiving packets at a virtual private network entity; and

building at least one long-term and at least one short-term statistical profile based on the received packets, and

comparing at least one long-term statistical profile with at least one short-term statistical profile to determine whether the packets indicate suspicious network activity.

26. The method of claim 25, further comprising decrypting the packets before statistically analyzing the packets.

27. The method of claim 25, further comprising not decrypting the packets before statistically analyzing the packets.

* * * * *

EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances †

Phillip A. Porras and Peter G. Neumann
porras@csl.sri.com and neumann@csl.sri.com
http://www.csl.sri.com/intrusion.html

Computer Science Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025-3493

Abstract— The EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) environment is a distributed scalable tool suite for tracking malicious activity through and across large networks. EMERALD introduces a highly distributed, building-block approach to network surveillance, attack isolation, and automated response. It combines models from research in distributed high-volume event-correlation methodologies with over a decade of intrusion detection research and engineering experience. The approach is novel in its use of highly distributed, independently tunable, surveillance and response monitors that are deployable polymorphically at various abstract layers in a large network. These monitors contribute to a streamlined event-analysis system that combines signature analysis with statistical profiling to provide localized real-time protection of the most widely used network services on the Internet. Equally important, EMERALD introduces a recursive framework for coordinating the dissemination of analyses from the distributed monitors to provide a global detection and response capability that can counter attacks occurring across an entire network enterprise. Further, EMERALD introduces a versatile application programmers' interface that enhances its ability to integrate with heterogeneous target hosts and provides a high degree of interoperability with third-party tool suites.

Keywords— Network security, intrusion detection, coördinated attacks, anomaly detection, misuse detection, information warfare, system survivability, insider threat, outsider threat.

I. Introduction

Our infrastructures of highly integrated information systems, both military and commercial, have become one of the key assets on which we depend for competitive advantage. These information infrastructures
1 tend to be conglomerates of integrated commercial off-the-shelf (COTS) and non-COTS components, interoperating and sharing information at increasing levels of 1
demand and capacity. These systems are relied on to manage a growing list of needs including transportation, commerce, energy management, communications, and defense.

Unfortunately, the very interoperability and sophisticated integration of technology that make our infras- 2
tructures such valuable assets also make them vulnerable to attack, and make our dependence on our infrastructures a potential liability. We have had ample opportunity to consider numerous examples of vulnerabilities and attacks against our infrastructures and the systems that use them. Attacks such as the Internet worm [21], [23] have shown us how our interconnectivity across large domains can be used against us to spread malicious code. Accidental outages such as the 3
1980 ARPAnet collapse [22] and the 1990 AT&T collapse [17] illustrate how seemingly localized triggering events can have globally disastrous effects on widely distributed systems. In addition, we have witnessed organized groups of miscreants [11], [17], local and foreign, performing malicious and coordinated attacks against varieties of online targets. We are keenly aware of the recurring examples of vulnerabilities that exist pervasively in network services, protocols, and operating systems, throughout our military and commercial network infrastructures. Even the deployment of newer more robust technologies does not fully compensate for the vulnerabilities in the multitude of legacy systems with which the newer systems must interoperate.

Yet, despite these examples, there remain no widely available robust tools to allow us to track malicious activity through and across large networks. The need for scalable network-aware surveillance and response technologies continues to grow.

† The work described here is currently funded by DARPA/ITO under contract number F30602-96-C-0294.

II. Challenges to Scalable Network Misuse Detection

As dependence on our network infrastructures continues to grow, so too grows our need to ensure the survivability of these assets. Investments into scalable network intrusion detection[1] will over time offer an important additional dimension to the survivability of our infrastructures. Mechanisms are needed to provide real-time detection of patterns in network operations that may indicate anomalous or malicious activity, and to respond to this activity through automated countermeasures. In addition, these mechanisms should also support the pursuit of individuals responsible for malicious activity through the collection and correlation of event data.

The typical target environment of the EMERALD project is a large enterprise network with thousands of users connected in a federation of independent administrative *domains*. Each administrative domain is viewed as a collection of local and network services that provide an interface for requests from individuals internal and external to the domain. Network services include features common to many network operating systems such as mail, HTTP, FTP, remote login, network file systems, finger, Kerberos, and SNMP. Some domains may share trust relationships with other domains (either peer-to-
4 peer or hierarchical). Other domains may operate in complete mistrust of all others, providing outgoing connections only, perhaps severely restricting incoming connections. Users may be local to a single domain or may possess accounts on multiple domains that allow them to freely establish connections throughout the enterprise.

In the environment of an enterprise network, well-established concepts in computer security such as the *reference monitor* [3] do not apply well. A large enterprise network is a dynamic cooperative of interconnected heterogeneous systems that often exists more through co-dependence than hierarchical structure. Defining a single security policy over such an enterprise, let alone a single point of authority, is often not practical.

With traditional approaches to security being difficult to apply to network infrastructures in the large, the need to ensure survivability of these infrastructures raises important questions. One such question is, *"Can we build surveillance and response capabilities that can scale to very large enterprise networks?"* To do so will require us to overcome a number of challenges in current intrusion-detection designs, many of which derive from the centralized paradigm of current architectures. While a fully distributed architecture could address some of these challenges, it too introduces tradeoffs in capabilities and performance. The following briefly summarizes challenges that exist in scaling intrusion-detection tools to large networks.

[1] In this paper, the term "intrusion" is used broadly to encompass misuse, anomalies, service denials, and other deviations from acceptable system behavior.

- **Event Generation and Storage:** Audit generation and storage has tended to be a centralized activity, and often gathers excessive amounts of information at inappropriate layers of abstraction. Centralized audit mechanisms place a heavy burden on the CPU and I/O throughput, and simply do not scale well with large user populations. In addition, it is difficult to extend centralized audit mechanisms to cover spatially distributed components such as network infrastructure (e.g., routers, filters, DNS, firewalls) or various common network services.

- **State-space Management and Rule Complexity:** In signature-based analyses, rule complexity can have a direct tradeoff with performance. A sophisticated rule structure able to represent complex/multiple event orderings with elaborate pre- or post-conditions may allow for very concise and well-structured penetration definitions. However, sophisticated rule structures may also impose heavy burdens in maintaining greater amounts of state information throughout the analysis, limiting their scalability to environments with high volumes of events. Shorter and simpler rules may impose lesser analysis and state-management burdens, helping to provide greater scalability and efficiency in event analysis. When speed is the key issue, the ultimate rule-set is one with no state-management needs — requiring no ordering and no time-consuming pre- and post-conditions to evaluate as events are processed. Simpler rules, however, also limit expressibility in misuse definitions, and can lead to inflated rule-bases to compensate for a single complex rule-set that might cover many variations of an attack. Clearly, there exists a tradeoff between highly complex and expressibly rich rule models versus shorter and simpler rules that individually require minimal state-management and analysis burdens.

- **Knowledge Repositories:** Expert systems separate their base of knowledge (rules of inference and state information regarding the target system) from both their analysis code and response logic in an effort to add to their overall modularity. There is some advantage to maintaining this knowledge base in a centrally located repository. Dynamic modification and control over this information is made easier when only single repositories need be modified. A centrally located knowledge repository is efficient for making pluggable rule-sets that add

to the generality and portability of the tool. However, in a highly distributed and high-volume event environment, a single repository combined with a single analysis engine can act as a choke-point. It also provides a single point of failure should the repository become unavailable or tainted.

- **Inference Architectures:** At the core of many signature-based expert systems exists an algorithm for accepting the input (in our case activity logs) and, based on a set of inference rules, directing the search for new information. This inference-engine model is very centralized in nature. In a large network, events and data flow asynchronously throughout the network in parallel and in volumes beyond what any centralized analysis technologies can process. A central analysis requires centralized collection of event information, and imposes the full burden (I/O, processing, and memory) of the analysis on those components on which the inference engine resides. This single-point-of-analysis model does not scale well. A completely distributed analysis, however, introduces its own challenges. Both global correlation and intelligent coordination among distributed analysis units impose significant resource overhead. Finding the optimal analysis paradigm between the continuum of the centralized expert-system approach and a fully decentralized analysis scheme is a key challenge in building a scalable inference architecture.

The physical and logical dispersion of the interfaces and controls among target systems and networks must be accommodated by the architecture of the distributed analysis system. Centralized intrusion-detection architectures deployed in highly distributed network environments experience difficulty in integrating and scaling their analysis paradigms to such environments. (Several of these issues are explored in [16]). The issues and limitation discussed above represent challenges to the very design and engineering assumptions on which much of the current intrusion-detection research is based.

The objective of the EMERALD work is to bring a collection of research and prototype development efforts into the practical world, in such a way that the analysis tools for detecting and interpreting anomalies and misuses can be applied and integrated into realistic network computing environments. The EMERALD project provides a critical step in demonstrating how to construct scalable and computationally realistic intrusion-detection mechanisms to track malicious activity within and across large networks. To do this, EMERALD employs detection and response components that are smaller and more distributed than previous intrusion-detection efforts, and that interoperate to provide composable surveillance.

EMERALD represents a significant departure from previous centralized host-based, user-oriented, intrusion-detection efforts that suffer poor scalability and integration into large networks. EMERALD's analysis scheme targets the external threat agent who attempts to subvert or bypass a domain's network interfaces and control mechanisms to gain unauthorized access to domain resources or prevent the availability of these resources. EMERALD employs a building-block architectural strategy using independent distributed surveillance monitors that can analyze and respond to malicious activity on local targets, and can interoperate to form an analysis hierarchy. This layered analysis hierarchy provides a framework for the recognition of more global threats to interdomain connectivity, including coordinated attempts to infiltrate or destroy connectivity across an entire network enterprise. Section III presents an architectural overview of EMERALD, and Section IV discusses its integration into distributed computing environments. 5

III. The EMERALD Network Intrusion Detection Architecture

EMERALD introduces a hierarchically layered approach to network surveillance that includes *service analysis* covering the misuse of individual components and network services within the boundary of a single domain; *domain-wide analysis* covering misuse visible across multiple services and components; and *enterprise-wide analysis* covering coordinated misuse across multiple domains. The objective of the service analysis is to streamline and decentralize the surveillance of a domain's network interfaces for activity that may indicate misuse or significant anomalies in operation. We introduce the concept of dynamically deployable highly distributed, and independently tunable *service monitors*. Service monitors are dynamically deployed within a domain to provide localized real-time analysis of infrastructure (e.g., routers or gateways) and services (privileged subsystems with network interfaces). Service monitors may interact with their environment passively (reading activity logs) or actively via probing to supplement normal event gathering. This localized coverage of network services and domain infrastructure forms the lowest tier in EMERALD's layered network-monitoring scheme. 6

Information correlated by a service monitor can be disseminated to other EMERALD monitors through a *subscription*-based communication scheme. Subscription provides EMERALD's message system both a push and pull data exchange capability between monitor interoperation (see Section III-F). EMERALD client monitors are able to subscribe to receive the analysis 7

esults that are produced by server monitors. As a monitor produces analysis results, it is then able to disseminate these results asynchronously to its client subscribers. Through subscription, EMERALD monitors distributed throughout a large network are able to efficiently disseminate reports of malicious activity without requiring the overhead of synchronous polling.

Domain-wide analysis forms the second tier of EMERALD's layered network surveillance scheme. A *domain monitor* is responsible for surveillance over all or part of the domain. Domain monitors correlate intrusion reports disseminated by individual service mon-
6 itors, providing a domain-wide perspective of malicious activity (or patterns of activity). In addition to domain surveillance, the domain monitor is responsible for reconfiguring system parameters, interfacing with other monitors beyond the domain, and reporting threats against the domain to administrators.

Lastly, EMERALD enables enterprise-wide analysis, providing a global abstraction of the cooperative com-
10 munity of domains. Enterprise-layer monitors correlate activity reports produced across the set of monitored domains. Enterprise-layer monitors focus on network-wide threats such as Internet worm-like attacks, attacks repeated against common network services across domains, and coordinated attacks from multiple domains against a single domain. Through this correlation and sharing of analysis results, reports of problems found by one monitor may propagate to other monitors through-
12 out the network. The enterprise itself need not be stable in its configuration or centrally administered. Rather, it may exist as an emergent entity through the interconnections of the domains. EMERALD's ability to perform interdomain event analysis is vital to addressing more global, information warfare-like attacks against the entire enterprise (see Section IV).

A. The EMERALD Monitor

The generic EMERALD monitor architecture is illustrated in Figure 1. The architecture is designed to enable the flexible introduction and deletion of analysis engines from the monitor boundary as necessary. In
14 its dual-analysis configuration, an EMERALD monitor instantiation combines signature analysis with statistical profiling to provide complementary forms of analysis over the operation of network services and infrastructure. In general, a monitor may include additional analysis engines that may implement other forms of event analysis, or a monitor may consist of only a single resolver implementing a response policy based on intrusion summaries produced by other EMERALD monitors. Monitors also incorporate a versatile application programmers' interface that enhances their ability to interoperate with the analysis target, and with other third-party intrusion-detection tools.

Underlying the deployment of an EMERALD monitor is the selection of a target-specific event stream. The event stream may be derived from a variety of sources 15
including audit data, network datagrams, SNMP traffic, application logs, and analysis results from other intrusion-detection instrumentation. The event stream is parsed, filtered, and formatted by the target-specific event-collection methods provided within the resource object definition (see Section III-B). Event records are 18
then forwarded to the monitor's analysis engine(s) for processing.

EMERALD's *profiler engine* performs statistical profile-based anomaly detection given a generalized event stream of an analysis target (Section III-C). EMERALD's *signature engine* requires minimal state-management and employs a rule-coding scheme that breaks from traditional expert-system techniques to provide a more focused and distributed signature-analysis model (Section III-D). Multiple analysis engines implementing different analysis methods may be employed to analyze a variety of event streams that pertain to the same analysis target. These analysis engines are intended to develop significantly lower volumes of abstract *intrusion* or *suspicion reports*. The profiler and signature engines receive large volumes of event logs 21
specific to the analysis target, and produce smaller volumes of intrusion or suspicion reports that are then fed to their associated *resolver*.

EMERALD's resolver is the coordinator of analysis reports and the implementor of the "response policy" (Section III-E). A resolver may correlate analysis results produced externally by other analysis engines to which it subscribes, and it may be bound to one or more analysis engines within the monitor boundary. Because the volume of its input is much lower than the event-stream volumes processed by the analysis engines, the resolver is able to implement sophisticated management and control policies over the analysis engines. The resolver also provides the primary interface between its associated analysis engines, the analysis target, and other intrusion-detection modules. In general, monitors may exist with multiple analysis engines, and support the capability to interoperate with third-party analysis engines.

At the center of the EMERALD monitor is a structure called a *resource object*. The resource object is a pluggable library of target-specific configuration data and methods that allows the monitor code-base to remain independent from the analysis target to which it is deployed (Section III-B). Customizing and dynamically configuring an EMERALD monitor thus becomes

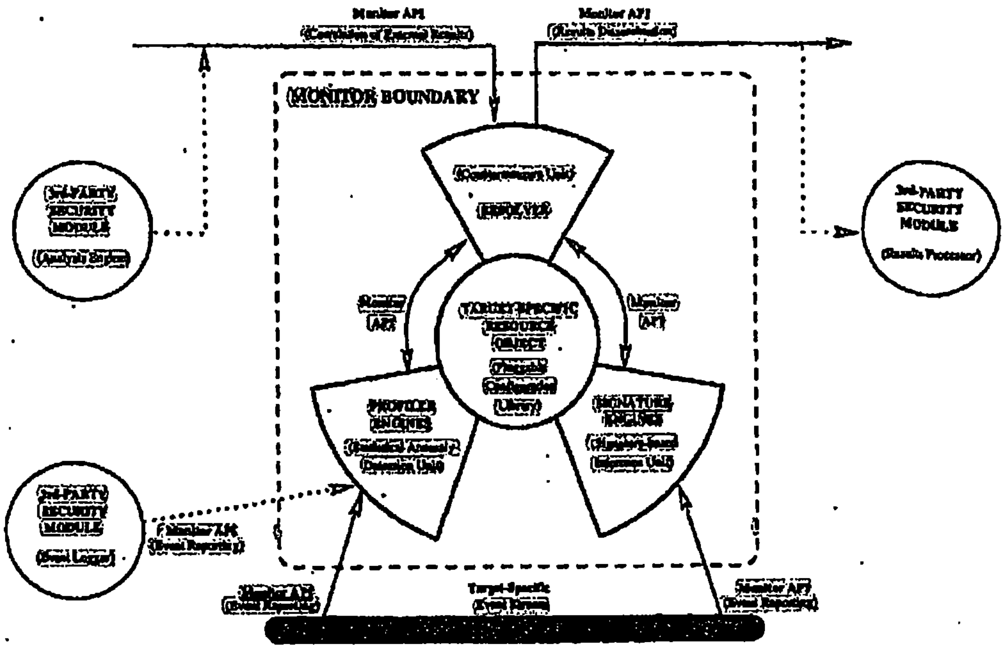


Fig. 1. The Generic EMERALD Monitor Architecture

a question of building and defining the fields of the analysis target's resource object.

Interoperability is especially critical to EMERALD's decentralized monitoring scheme, and extends within EMERALD's own architectural scope as well as to third-party modules. To support interoperability, EMERALD monitors incorporate a bidirectional messaging system. Section III-F discusses our efforts to develop a standard interface specification for communication within and between EMERALD monitors and

28 external modules. Using this interface specification, third-party modules can communicate with EMERALD monitors in a variety of ways, as illustrated in Figure 1. Third-party modules operating as event-collection units may employ EMERALD's external interfaces to submit event data to the analysis engines for processing. Such third-party modules would effectively replace the monitor's own event-collection methods (Section III-B). Third-party modules may also submit and receive analysis results via the resolver's external interfaces. This will allow third-party modules to incorporate the results from EMERALD monitors into their own surveillance efforts, or to contribute their results to the EMERALD analysis hierarchy. Lastly, the monitor's internal API

29 allows third-party analysis engines to be linked directly into the monitor boundary.

All EMERALD monitors (service, domain, and en- 38
terprise) are implemented using the same monitor code-base. The EMERALD monitor architecture is designed generally enough to be deployed at various abstract layers in the network. The only differences between deployed monitors are their resource object definitions. This reusable software architecture is a major project asset, providing significant benefits to the implementation and maintenance efforts. The following sections briefly describe the various components that make up the EMERALD monitor architecture.

B. *Resource Objects: Abstracting Network Entities*

Fundamental to EMERALD's design is the abstraction of the semantics of the analysis target from the EMERALD monitor. By logically decoupling the implementation of the EMERALD monitor from the analysis semantics of the analysis target, the extension of EMERALD's surveillance capabilities becomes a question of integration rather than implementation. The resource object contains all the operating parameters for each of the monitor's components as well as the analysis
semantics (e.g., the profiler engine's measure and cat- 39
egory definition, or the signature engine's penetration rule-base) necessary to process the target event stream. Once the resource object for a particular analysis target

is defined, it may be reused later by other EMERALD
39 monitors that are deployed to equivalent analysis targets. For example, the resource object for a domain's router may be reused as other EMERALD monitors are deployed for other routers in the domain. A library of resource object definitions is being developed for commonly available network surveillance targets.

Figure 2 illustrates the general structure of the resource object. The resource object provides a pluggable
40 configuration module for tuning the generic monitor code-base to a specific analysis target event stream. It minimally comprises the following variables (these variables may be extended as needed to accommodate the incorporation of new analysis engines into the monitor boundary):

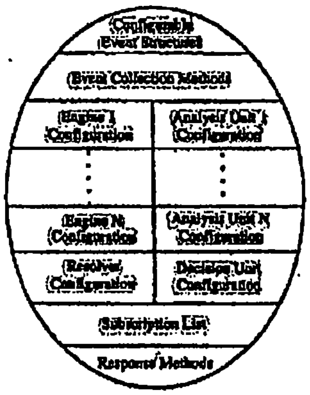


Fig. 2. The Generic EMERALD Monitor Architecture

- **Configurable Event Structures:** The monitor code-base maintains no internal dependence on the content or format of any given target event stream or the analysis results produced from analyzing the event stream. Rather, the resource object provides a univer-
41 sally applicable syntax for specifying the structure of event records and analysis results. Event records are defined based on the contents of the monitor's target event stream(s). Analysis result structures are used to package the findings produced by the analysis engine. Event records and analysis results are defined similarly to allow the eventual hierarchical processing of analysis results as event records by subscriber monitors.

- **Event-Collection Methods:** A set of filtering routines (or log conversion routines with custom filtering semantics) is employed by the analysis engines to gather and format target-specific event records. These are the native methods that interact directly with the system to parse the target event stream.

- **Engine N Configuration:** This refers to a collection of variables and data structures that specifies the operating configuration of a fielded monitor's analysis engine(s). The resource object maintains a separate collection of operating parameters for each analysis engine instantiated within the monitor boundary.

- **Analysis Unit N Configuration:** Each analysis engine maintains an independently configured collection of intrusion-detection analysis procedures. This structure contains the configuration variables that define the semantics employed by the analysis engine to process the target-specific event stream.

- **Resolver Configuration:** The resource object 42 maintains the operating parameters that specify the configuration of the resolver's internal modules.

- **Decision Unit Configuration:** This refers to the semantics used by the resolver's decision unit for 43 merging the analysis results from the various analysis engines. The semantics include the response criteria used by the decision unit for invoking countermeasure handlers.

- **Subscription List:** This structure contains information necessary for establishing subscription-based 44 communication sessions, which may include network address information and public keys used by the monitor to authenticate potential clients and servers. The subscription list field is an important facility for gaining visibility into malicious or anomalous activity outside the immediate environment of an EMERALD monitor. The most obvious examples where relationships are important involve interdependencies among network services that make local policy decisions. Consider, for 45 example, the interdependencies between access checks performed during network file system mounting and the IP mapping of the DNS service. An unexpected mount monitored by the network file system service may be responded to differently if the DNS monitor informs the network file system monitor of suspicious updates to the mount requestor's DNS mapping.

- **Valid Response Methods:** Various response functions can be made available to the resolver as it receives intrusion reports from its analysis engines or intrusion summaries from subscribees. These are preprogrammed countermeasure methods that the resolver may invoke as intrusion summaries are received.

As discussed above, the fields of the resource object are defined and utilized during monitor initialization. 46 In addition, these fields may be modified by internal monitor components and by authorized external clients using the monitor's API. Once fields are modified, components can be requested to dynamically reload the configuration parameters defined in those fields. This gives EMERALD an important ability to provide adaptive

analysis a control functionality. However, it also introduces a potential stability problem if dynamic modifications are not tightly restricted to avoid cyclic modifications. To address this issue, monitors accept configuration requests from only immediate parents in EMERALD's analysis hierarchy.

C. Scalable Profile-Based Anomaly Detection

The original groundwork for SRI's IDES effort was performed over a decade ago. The first-generation statistics component was used to analyze System Management Facility (SMF) records from an IBM mainframe system [10] in the first half of the 1980s. Requirements for an anomaly-detection system that became IDES were documented in [6]. This research led to the development of the NIDES statistical profile-based anomaly-detection subsystem (NIDES/Stats), which employed a wide range of multivariate statistical measures to profile the behavior of individual users [9]. Analysis is user-based, where a statistical score is assigned to each user's session representing how closely currently observed usage corresponds to the established patterns of usage for that individual. The input source to the NIDES statistical component is an unfiltered and unsorted host audit log, which represents the activity of all users currently operating on the host.

In 1995, SRI conducted research under Trusted Information Systems' Safeguard project to extend NIDES/Stats to profile the behavior of individual applications [2]. Statistical measures were customized to measure and differentiate the proper operation of an application from operation that may indicate Trojan horse substitution. Under the Safeguard model, analysis is application-based, where a statistical score is assigned to the operation of applications and represents the degree to which current behavior of the application corresponds to its established patterns of operation. The Safeguard effort demonstrated the ability of statistical profiling tools to clearly differentiate the scope of execution among general-purpose applications. It also showed that statistical analyses can be very effective in analyzing activities other than individual users; by instead monitoring applications, the Safeguard analysis greatly reduced the required number of profiles and computational requirements, and also dramatically decreased the typical false-positive and false-negative ratios.

While NIDES/Stats has been reasonably successful profiling users and later applications, it will be extended to the more general subject class typography required by EMERALD. Nonetheless, the underlying mechanisms are well suited to the problem of network anomaly detection, with some adaptation. The required modifications center around extensive reworking of NIDES/Stats to abstract and generalize its definition of measures and profiles, the streamlining of its profile management, and the adaptation of the configuration and reporting mechanisms to EMERALD's highly interoperable and dynamic message system interface.

The EMERALD profiler engine achieves total separation between profile management and the mathematical algorithms used to assess the anomaly of events. Profiles are provided to the computational engine as classes defined in the resource object. The mathematical functions for anomaly scoring, profile maintenance, and updating function in a fully general manner, not requiring any underlying knowledge of the data being analyzed beyond what is encoded in the profile class. The event-collection interoperability supports translation of elementary data (the analysis target's event stream) to the profile and measure classes. At that point, analysis for different types of monitored entities is mathematically similar. This approach imparts great flexibility to the analysis in that fading memory constants, update frequency, measure type, and so on are tailored to the entity being monitored. 17 18

Each profiler engine is dedicated to a specific target event stream at the elementary level. Such localized, target-specific analyses (unlike the monolithic approach employed by NIDES/Stats) provide a more distributed, building-block approach to monitoring, and allow profiling computations to be efficiently dispersed throughout the network. Because the event stream submitted to the profiler engine is specific to the analysis target's activity, profile management is greatly simplified, in that there is no need to support multisubject profile instantiations.

In addition, the results of service-layer profiler engines can be propagated to other monitors operating higher in EMERALD's layered analysis scheme, offering domain- or enterprise-wide statistical profiling of anomaly reports. Profiler engines may operate throughout the analysis hierarchy, further correlating and merging service-layer profiles to identify more widespread anomalous activity. The underlying mathematics are the same for each instance, and all required information specific to the entity being monitored (be it a network resource or other EMERALD monitors producing analysis results at lower layers in the analysis hierarchy) is entirely encapsulated in the objects of the profile class.

D. Scalable Signature Analysis

Signature analysis is a process whereby an event stream is mapped against abstract representations of event sequences that are known to indicate undesirable activity. However, simplistic event binding alone may not necessarily provide enough indication to ensure the 19

accurate detection of the target activity. Signature analyses must also distinguish whether an event sequence being witnessed is actually transitioning the system into the anticipated compromised state. Additionally, determining whether a given event sequence is indicative of an attack may be a function of the preconditions under which the event sequence is performed. To enable this finer granularity of signature recognition, previous efforts have employed various degrees of state detection and management logic (one such example is found in [18]). However, as discussed in Section II, the incorporation of sophisticated rule- and state-management features must be balanced with the need to ensure an acceptable level of performance.

In many respects, EMERALD's signature-analysis strategy departs from previous centralized rule-based efforts. EMERALD employs a highly distributed analysis strategy that, with respect to signature analysis, effectively modularizes and distributes the rule-base and inference engine into smaller, more focused signature engines. This has several benefits beyond the performance advantages from evenly distributing the computational load across network resources.

By narrowing the scope of activity in the event stream to a single analysis target, the noise ratio from event records that the signature engine must filter out is greatly reduced. This noise filtering of the event stream helps the signature engine avoid misguided searches along incorrect signature paths. EMERALD also partitions and distributes the signature activity representations. Rather than maintaining a central knowledge-base containing representations of all known malicious activity across a given computing environment, EMERALD distributes a tailored set of signature activity with each monitor's resource object.

47 EMERALD's signature-analysis objectives depend on which layer in EMERALD's hierarchical analysis scheme the signature engine operates. Service-layer signature engines attempt to monitor network services and infrastructure for attempts to subvert or misuse these components to penetrate or interfere with the domain's operation. Service monitors target external and perhaps unauthenticated individuals who attempt to subvert services or domain components to perform actions outside their normal operating scope. The EMERALD signature engine scans the event stream for events that represent attempted exploitations of known attacks against the service, or other activity that stands alone as warranting a response from the EMERALD monitor.

20 Above the service layer, signature engines scan the aggregate of intrusion reports from service monitors in an attempt to detect more global coordinated attack scenarios or scenarios that exploit interdependencies among network services. The DNS/NFS attack discussed in Section III-B is one such example of an aggregate attack scenario. The fault-propagation model presented in [20] offers a general example of modeling interdependency of network assets (in this case fault interdependencies in a nonmalicious environment) that is also of general relevance for EMERALD's domain- and enterprise-layer intrusion correlation.

E. A Universal Resolver: Correlation and Response

EMERALD maintains a well-defined separation between analysis activities and response logic. Implementation of the response policy, including coordinating the dissemination of the analysis results, is the responsibility of the EMERALD resolver. The resolver is an expert system that receives the intrusion and suspicion reports produced by the profiler and signature engines, and based on these reports invokes the various response handlers defined within the resource object. Because the volume of intrusion and suspicion reports is lower than the individual event reports received by the analysis engines, the resolver can afford the more sophisticated demands of maintaining the configuration, and managing the response handling and external interfaces necessary for monitor operation. Furthermore, the resolver adds to the extensibility of EMERALD by providing the subscription interface through which third-party analysis tools can interact, and participate in EMERALD's layered analysis scheme. 22

Upon its initialization, the resolver references various fields within the associated resource object. The resolver initiates authentication and subscription sessions with those EMERALD monitors whose identities appear in the resource object's subscription-list field. It also handles all incoming requests by subscribers, which must authenticate themselves to the resolver. (Details of EMERALD's subscription-session authentication process are discussed in [19].) Once a subscription session is established with a subscriber monitor, the resolver acts as the primary interface through which configuration requests are received, probes are handled, and intrusion reports are disseminated. 23

EMERALD supports extensive intermonitor sharing of analysis results throughout its layered analysis architecture. Resolvers are able to request and receive intrusion reports from other resolvers at lower layers in the analysis hierarchy. As analysis results are received from subscribees, they are forwarded via the monitor's event filters to the analysis engines. This tiered collection and correlation of analysis results allows EMERALD monitors to represent and profile more global malicious or anomalous activity that is not visible from the local monitoring of individual network services and assets 24

(see Section IV).

[25] In addition to its external-interface responsibilities, the resolver operates as a fully functional decision engine, capable of invoking real-time countermeasures in response to malicious or anomalous activity reports produced by the analysis engines. Countermeasures are defined in the response-methods field of the resource object. Included with each valid response method are evaluation metrics for determining the circumstances under which the method should be dispatched. These response criteria involve two evaluation metrics: a threshold metric that corresponds to the measure values and scores produced by the profiler engine, and severity metrics correspond to subsets of the associated attack sequences defined within the resource object. The resolver combines the metrics to formulate its monitor's response policy. Aggressive responses may include direct countermeasures such as closing connections or terminating processes. More passive responses may include the dispatching of integrity-checking handlers to verify the operating state of the analysis target.

[26] The resolver operates as the center of intramonitor communication. As the analysis engines build intrusion and suspicion reports, they propagate these reports to the resolver for further correlation, response, and dissemination to other EMERALD monitors. The resolver can also submit runtime configuration requests to the analysis engines, possibly to increase or decrease the scope of analyses (e.g., enable or disable additional signature rules) based on various operating metrics. These configuration requests could be made as a result of encountering other intrusion reports from other subscribers. For example, an intrusion report produced by a service monitor in one domain could be propagated to an enterprise monitor, which in turn sensitizes service monitors in other domains to the same activity.

[27] Lastly, a critical function of the EMERALD resolver is to operate as the interface mechanism between the monitor administrator and the monitor itself. From the perspective of an EMERALD resolver, the administrator interface is simply a subscribing service to which the resolver may submit its intrusion summaries and receive probes and configuration requests. The administrative interface tool can dynamically subscribe and unsubscribe to any of the deployed EMERALD resolvers, as well as submit configuration requests and asynchronous probes as desired.

F. The EMERALD Message System

Interoperability is especially critical to the EMERALD design, which from conception promotes dynamic extensibility through a building-block approach to scalable network surveillance. EMERALD monitors incorporate a duplex messaging system that allows them to correlate activity summaries and countermeasure information in a distributed hierarchical analysis framework. EMERALD's messaging system must address interoperability both within its own architectural scope and with other third-party analysis tools. To do this, the messaging system provides a well-defined programmer's interface that supports the bidirectional exchange of analysis results and configuration requests with alternative security tools.

EMERALD's message system operates under an asynchronous communication model for handling results dissemination and processing that is generically referred to as subscription-based message passing.[2] EMERALD component interoperation is client/server-based, where a client module may subscribe to receive event data or analysis results from servers. Once the subscription request is accepted by the server, the server module forwards events or analysis results to the client automatically as data becomes available, and may dynamically reconfigure itself as requested by the client's control requests. While this asynchronous model does not escape the overhead needed to ensure reliable delivery, it does reduce the need for client probes and acknowledgments. [30]

An important goal in the design of EMERALD's interface specification is that the interface remain as implementation neutral as possible. To support an implementation-neutral communication framework, the message system is designed with strong separation between the programmer's interface specification and the issues of message transport.[3] The interface specification embodies no assumptions about the target intrusion-detection modules, implementation languages, host platform, or network. The transport layer is architecturally isolated from the internals of EMERALD monitors so that transport modules may be readily introduced and replaced as protocols and security requirements are negotiated between module developers. The following briefly summarizes EMERALD's interface specification and transport layer design. [31]

Interface Specification: Interface specification involves the definition of the messages that the various intrusion-detection modules must convey to one another, and how these messages should be processed. The message structure and content are specified in a completely implementation-neutral context. Internally, EMERALD monitors contain three general module types: event collection methods that collect and fil- [32]

[2] Other communities have employed subscription-based push/pull data flow schemes for information such as network management traffic and WWW content.

[3] Details of EMERALD's programmer's interface specification and transport design are provided in [19].

ter the target event stream, analysis engines that process the filtered events, and a resolver that processes and responds to the analysis engine results. Externally, EMERALD monitors interoperate with one another in a manner analogous to internal communication: service monitors produce local analysis results that are passed to the domain monitor; domain monitors correlate service monitor results, producing new results that are further propagated to enterprise monitors; enterprise monitors correlate and respond to the analysis results produced by domain monitors.

Both intramonitor and intermonitor communication employ identical subscription-based client-server models. With respect to intermonitor communication, the resolver operates as a client to the analysis engines, and the analysis engines operate as clients to the event filters. Through the internal message system, the resolver [33] submits configuration requests and probes to the analysis engines, and receives from the analysis engines their analysis results. The analysis engines operate as servers providing the resolver with intrusion or suspicion reports either asynchronously or upon request. Similarly, the analysis engines are responsible for establishing and maintaining a communication link with a target event collection method (or event filter) and prompting the reconfiguration of the collection method's filtering semantics when necessary. Event collection methods provide analysis engines with target-specific event records upon which the statistical and signature analyses are performed.

Intermonitor communication also operates using the subscription-based hierarchy. A domain monitor subscribes to the analysis results produced by service monitors, and then propagates its own analytical results to its parent enterprise monitor. The enterprise monitor operates as a client to one or more domain monitors, [34] allowing them to correlate and model enterprise-wide activity from the domain-layer results. Domain monitors operate as servers to the enterprise monitors, and as clients to the service-layer monitors deployed throughout their local domain. This message schema would operate identically if correlation were to continue at higher layers of abstraction beyond enterprise analysis.

EMERALD's intramonitor and intermonitor programming interfaces are identical. These interfaces are subdivided into five categories of interoperation: channel initialization and termination, channel synchroniza- [35] tion, dynamic configuration, server probing, and report/event dissemination. Clients are responsible for initiating and terminating channel sessions with servers. Furthermore, clients are responsible for managing channel synchronization in the event of errors in message sequencing or periods of failed or slow response (i.e., "I'm alive" confirmations). Clients may also submit dynamic configuration requests to servers. For example, an analysis engine may request an event collection method to modify its filtering semantics. Clients may [35] also probe servers for report summaries or additional event information. Lastly, servers may send clients intrusion/suspicion summaries or event data in response to client probes or in an asynchronous dissemination mode.

Transport Layer: The second part of the message system framework involves the specification of the transport mechanism used to establish a given communication channel between monitors or possibly between a monitor and a third-party security module. All im- [36] plementation dependencies within the message system framework are addressed by the pluggable transport modules. Transport modules are specific to the participating intrusion-detection modules, their respective hosts, and potentially to the network—should the modules require cross-platform interoperation. Part of the integration of a monitor into a new analysis target is the incorporation of the necessary transport module(s) (for both internal and external communication).

It is at the transport layer where EMERALD addresses issues of communications security, integrity, and reliability. While it is important to facilitate interoperability among security mechanisms, this interoperability must be balanced with the need to ensure an overall level of operational integrity, reliability, and privacy. An essential element in the EMERALD messaging system design is the integration of secure transport to ensure a degree of internal security between EMERALD components and other cooperative analysis units.

The transport modules that handle intramonitor communication may be different from the transport modules that handle intermonitor communication. This allows the intramonitor transport modules to address security and reliability issues differently than how the intermonitor transport modules address security and reliability. While intramonitor communication may more commonly involve interprocess communication within a single host, intermonitor communication will most commonly involve cross-platform networked interopera- [37] tion. For example, the intramonitor transport mechanisms may employ unnamed pipes [14], which provides a kernel-enforced private interprocess communication channel between the monitor components (this assumes a process hierarchy within the monitor architecture). The monitor's external transport, however, will more likely export data through untrusted network connections and thus require more extensive security management. To ensure the security and integrity of the message exchange, the external transport may employ

public/private key authentication protocols and session
37 key exchange. Using this same interface, third-party analysis tools may authenticate and exchange analysis results and configuration information with EMERALD monitors in a well-defined, secure manner.

The pluggable transport allows EMERALD flexibility in negotiating security features and protocol usage with third parties. Of particular interest to the monitoring of network events is our planned incorporation of a commercially available network management system as a third-party module. That system will deliver monitoring results relating to security, reliability, availability, performance, and other attributes. The network management system may in turn subscribe to EMERALD results in order to influence network reconfiguration. This experiment will demonstrate the interoperation of intrusion-detection instrumentation with analysis tools that themselves do not specifically address security management.

IV. EMERALD Network Deployment

The EMERALD reusable-monitor architecture provides a framework for the organization and coordination of distributed event analysis across multiple administrative domains. EMERALD introduces a service-oriented, layered approach to representing, analyzing, and responding to network misuse. EMERALD's profiling and signature analyses are not performed as monolithic analyses over an entire domain, but rather are deployed sparingly throughout a large enterprise to provide focused protection of key network assets vulnerable to attack. This model leads to greater flexibility whenever the network configuration changes dynamically, and to improved performance, where computational load is distributed efficiently among network resources.

Domains under EMERALD surveillance are able to detect malicious activity targeted against their network services and infrastructure, and disseminate this information in a coordinated and secure way to other EMERALD monitors (as well as third-party analysis tools) distributed throughout the network. Reports of problems found in one domain can propagate to other monitors throughout the network using the subscription process. EMERALD's subscription-based communication strategy provides mutual authentication between participants, as well as confidentiality and integrity for all intermonitor message traffic (see Section III-F).

EMERALD's analysis scheme is highly composable, beginning at the service layer where EMERALD monitors analyze the security-relevant activity associated with an individual network service or network infrastructure. As service-layer monitors detect activity that indicates possible misuse, this information is responded to by the monitor's local resolver to ensure immediate response. Misuse reports are also disseminated throughout EMERALD's web of surveillance, to the monitor's pool of subscribers.

Domain-layer monitors model and profile domain-wide vulnerabilities not detectable from the narrow visibility of the service layer. Domain monitors search for intrusive and anomalous activity across a group of interdependent service-layer components, subscribing to each service's associated service monitor. Domain monitors also operate as the dissemination point between the domain's surveillance and the external network surveillance. Where mutual trust among domains exists, domain monitors may establish peer relationships with one another. Peer-to-peer subscription allows domain monitors to share intrusion summaries from events that have
occurred in other domains. Domain monitors may use 9
such reports to dynamically sensitize their local service monitors to malicious activity found to be occurring outside the domain's visibility. Domain monitors may also operate within an enterprise hierarchy, where they disseminate intrusion reports to enterprise monitors for global correlation. Where trust exists between domains, peer-to-peer subscription provides a useful technique for keeping domains sensitized to malicious activity occurring outside their view.

Enterprise-layer monitors attempt to model and detect coordinated efforts to infiltrate domain perimeters or prevent interconnectivity between domains. Enter-
prise surveillance may be used where domains are in- 11
terconnected under the control of a single organization, such as a large privately owned WAN. Enterprise surveillance is very similar to domain surveillance: the *enterprise monitor* subscribes to various domain monitors, just as the domain monitors subscribed to various local service monitors. The enterprise monitor (or monitors, as it would be important to avoid centralizing any analysis) focuses on network-wide threats such as Internet worm-like attacks, attacks repeated against com- 13
mon network services across domains, or coordinated attacks from multiple domains against a single domain. As an enterprise monitor recognizes commonalities in intrusion reports across domains (e.g., the spreading of a worm or a mail system attack repeated throughout the enterprise), its resolver can take steps to help domains counter the attack, and can also help sensitize other domains to such attacks before they are affected.

EMERALD's distributed analysis paradigm provides several significant performance advantages over the centralized signature analysis and statistical profiling tools from which its architecture is derived. In a large network, event activity is dispersed throughout its spa-

tially distributed components, occurring in parallel and in volumes that are difficult for centralized analysis tools to manage. EMERALD distributes the computational load and space utilization needed to monitor the various network components, and performs its analysis and response activity locally. Local detection and response also helps to ensure timely protection of network assets. Furthermore, EMERALD's distributed monitor deployment effectively parallelizes the statistical profiling and signature analyses. Once the event streams from the various analysis targets are separated and submitted to the deployed monitors, event correlation, profiling, and response handling are all managed by independent computational units. Lastly, EMERALD's dynamic extensibility allows an integrator to selectively choose the key elements in a network that require monitoring, and the ability to alter analysis coverage dynamically.

V. Related Work

EMERALD is not intended as a replacement to more centralized, host-based, user-oriented intrusion-detection tools, but rather as a complementary architecture that addresses threats from the interconnectivity of domains in hostile environments. Specifically, EMERALD attempts to detect and respond to both anticipated and unanticipated misuses of services and infrastructure in large network-based enterprises, including external threats that attempt to subvert or bypass a domain's network interfaces and control mechanisms to gain unauthorized access to domain resources or prevent the availability of these resources. EMERALD also provides a framework for recognizing more global threats to interdomain connectivity, including coordinated attempts to infiltrate or destroy connectivity across an entire network enterprise. A more detailed discussion of EMERALD's relationship with other work is given in [19]. Here, we merely allude to its position in the spectrum of research in intrusion detection, fault detection, and alarm correlation.

A. Related Intrusion Detection Research

EMERALD considerably generalizes and extends the earlier pioneering work of SRI's IDES and NIDES [1], overcoming previous limitations with respect to scalability, applicability to networking, interoperability, and inability to detect distributed coordinated attacks. It generalizes to network environments the Safeguard experience [2], which overcame profile explosion and scalability problems by locally profiling the activities of subsystems and commands rather than of individual users. EMERALD also extends the statistical-profile model of NIDES, to analyze the operation of network services, network infrastructure, and activity reports from other EMERALD monitors. Various other efforts have considered one of the two types of analysis – signature-based (e.g., Porras [18] has used a state-transition approach; the U.C. Davis and Trident DIDS [4] addresses abstracted analysis for networking, but not scalability; the Network Security Monitor [7] seeks to analyze packet data rather than conventional audit trails; Purdue [5] seeks to use adaptive-agent technology) or profile-based. More recent work in UC Davis' GrIDS effort [24] employs *activity graphs* of network operations to search for traffic patterns that may indicate network-wide coordinated attacks. (Ko has considered writing specifications for expected behavior [13], which is sort of a compromise between signature analysis and behavioral profiling.)

B. Related Research in Fault Detection

EMERALD is somewhat similar conceptually to various efforts in alarm correlation and high-volume event correlation/fault detection in the network management community [8], [15], [16]. EMERALD's architecture and layered analysis is somewhat similar to the distributed event correlation system (DECS) discussed in [12]. However, DECS makes several simplifications in its stateless event modeling scheme that do not translate well to a malicious environment for detecting intrusions. Recent work in nonmalicious fault isolation [20] is also relevant, and is being considered. However, none of these efforts shares EMERALD's abilities for recursive hierarchical abstraction and misuse detection, nor do they include provisions to ensure their own survivability in hostile environments.

VI. Conclusions

This paper introduces EMERALD, a composable surveillance and response architecture oriented toward the monitoring of distributed network elements. EMERALD targets external threat agents who attempt to subvert or bypass network interfaces and controls to gain unauthorized access to domain resources. EMERALD builds a multiple local monitoring capability into a framework for coordinating the dissemination of distributed analyses to provide global detection and response to network-wide coordinated attacks. The basic analysis unit in this architecture is the EMERALD monitor, which incorporates both signature analysis and statistical profiling. By separating the analysis semantics from the analysis and response logic, EMERALD monitors can be easily integrated throughout EMERALD's layered network surveillance strategy.

EMERALD builds on and considerably extends past research and development in anomaly and misuse detection, to accommodate the monitoring of large dis-

tributed systems and networks. Because the real-time analysis itself can be distributed and applied where most effective at different layers of abstraction, EMERALD has significant advantages over more centralized approaches in terms of event detectability and response capabilities, and yet can be computationally realistic. It can detect not only local attacks, but also coordinated attacks such as distributed denials of service. The EMERALD design addresses interoperability within its own scope, and in so doing enables its interoperability with other analysis platforms as well. EMERALD's inherent generality and flexibility in terms of what is being monitored and how the analysis is accomplished suggests that the design can be readily extended to monitoring other attributes such as survivability, fault tolerance, and assured availability.

References

[1] D. Anderson, T. Frivold, and A. Valdes. Next-generation intrusion-detection expert system (NIDES). Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, SRI-CSL-95-07, May 1995.

[2] D. Anderson, T. Lunt, H. Javitz, A. Tamaru, and A. Valdes. Safeguard final report: Detecting unusual program behavior using the NIDES statistical component. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, 2 December 1993.

[3] J.P. Anderson. Computer security technology planning study. Technical Report ESD-TR-73-51, ESD/AFSC, Hanscom AFB, Bedford, MA, October 1972.

[4] J. Brentano, S.R. Snapp, G.V. Dias, T.L. Goan, L.T. Heberlein, C.H. Ho, K.N. Levitt, and B. Mukherjee. An architecture for a distributed intrusion detection system. In *Fourteenth Department of Energy Computer Security Group Conference*, pages 25–45 in section 17, Concord, CA, May 1991. Department of Energy.

[5] M. Crosbie and E.H. Spafford. Active defense of a computer system using autonomous agents. Technical report, Department of Computer Sciences, CSD-TR-95-008, Purdue University, West Lafayette IN, 1995.

[6] D.E. Denning and P.G. Neumann. Requirements and model for IDES – a real-time intrusion-detection expert system. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, August 1985.

[7] T.L Heberlein, B. Mukherjee, and K.N. Levitt. A method to detect intrusive activity in a networked environment. In *Proceedings of the Fourteenth National Computer Security Conference*, pages 362–371, Washington, D.C., 1–4 October 1991. NIST/NCSC.

[8] G. Jakobson and M.D. Weissman. Alarm correlation. *IEEE Network*, pages 52–59, November 1993.

[9] H.S. Javitz and A. Valdes. The NIDES statistical component description and justification. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, March 1994.

[10] H.S. Javitz, A. Valdes, D.E. Denning, and P.G. Neumann. Analytical techniques development for a statistical intrusion-detection system (SIDS) based on accounting records. Technical report, SRI International, Menlo Park, CA, July 1986.

[11] P.M. Joyal. Industrial espionage today and information wars of tomorrow. In *National Information Systems Security Conference. Baltimore, Maryland*, pages 139–150, Washington, D.C., 22–25 October 1996.

[12] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo. A coding approach to event correlation. In *Proceedings of the Fourth International Symposium on Integrated Network Management (IFIP/IEEE), Santa Barbara, CA, May 1995*, pages 266–277. Chapman & Hall, London, England, 1995.

[13] C. Ko, M. Ruschitzka, and K. Levitt. Execution monitoring of security-critical programs in distributed systems: A specification-based approach. In *Proceedings of the 1997 Symposium on Security and Privacy*, pages 175–187, Oakland, CA, May 1997. IEEE Computer Society.

[14] D. Lewine. *POSIX Programmer's Guide*. O'Reilly and Associates Incorporated, 1991.

[15] M. Mansouri-Samani and M. Sloman. Monitoring distributed systems. *IEEE Network*, pages 20–30, November 1993.

[16] K. Meyer, M. Erlinger, J. Betser, C. Sunshine, G. Goldszmidt, and Y. Yemini. Decentralizing control and intelligence in network management. In *Proceedings of the Fourth International Symposium on Integrated Network Management (IFIP/IEEE), Santa Barbara, CA, May 1995*, pages 4–16, Chapman & Hall, London, England, 1995.

[17] P.G. Neumann. *Computer-Related Risks*. ACM Press, New York, and Addison-Wesley, Reading, MA, 1994. ISBN 0-201-55805-X.

[18] P.A. Porras and R.A. Kemmerer. Penetration state transition analysis: A rule-based intrusion detection approach. In *Proceedings of the Eighth Annual Computer Security Applications Conference (San Antonio, TX, Nov.30-Dec.4)*, pages 220–229. IEEE, 1992.

[19] P.A. Porras and P.G. Neumann. Conceptual design and planning for EMERALD: event monitoring enabling responses to anomalous live disturbances. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, October 1997. Available for download via http://www.csl.sri.com/intrusion.html.

[20] L. Ricciulli and N. Shacham. Modelling correlated alarms in network management systems. *Communications Networks and Distributed Systems Modeling and Simulation*, 1997.

[21] J.A. Rochlis and M.W. Eichin. With microscope and tweezers: The Worm from MIT's perspective. *Communications of the ACM*, 32(6):689–698, June 1989.

[22] E. Rosen. Vulnerabilities of network control protocols. *ACM SIGSOFT Software Engineering Notes*, 6(1):6–8, January 1981.

[23] E.H. Spafford. The Internet Worm: crisis and aftermath. *Communications of the ACM*, 32(6):678–687, June 1989.

[24] S. Staniford-Chen, S. Cheung, R. Crawford, J. Frank M. Dilger, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkel. Grids—a graph based intrusion detection system for large networks. In *Proceedings of the Nineteenth National Information Systems Security Conference*, pages 361–370, October 1996.